

AD-A268 814



INSTITUTE FOR SIMULATION AND TRAINING



Contract Number N61339-89-C-0045
CDRL A006
November 25, 1991

DTIC
ELECTE
AUG 23 1993
S E D

Aviation Trainer Technology Test Plan

Volume II: Software Development

~~STRIPED STRIPED~~
Approved for public release
Distribution Unlimited



IST

Institute for Simulation and Training
12424 Research Parkway, Suite 300
Orlando FL 32826

University of Central Florida
Division of Sponsored Research

93-19468



IST-TR-90-25



Aviation Trainer Technology Test Plan

Volume II: Software Development

Contract Number N61339-89-0045
CDRL A006
November 25, 1991

IST-TR-90-25

Statement A per Telecon
Raymond Green STRICOM/AMSTI-EC
Orlando, FL 32826-3276

NWW - 23 Aug 93

Accession For	
NTIS CRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input checked="" type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

DTIC QUALITY INSPECTED 3

Prepared by

Kevin Uliano

Reviewed by

Brian Goldiez

Table of Contents

<u>Section</u>	<u>Page</u>
List of Figures.	ii
1.0 Introduction	1
1.1 Intersimulator Network Design Description . . .	1
2.0 Software Introduction	4
3.0 System Header Files	5
3.1 Inet.h	5
3.2 Delay.h	8
3.3 Kklib.h	9
4.0 Clib.c	12
5.0 Intersim.c	87
6.0 Initsys.c	114
7.0 Predelay.c	119
8.0 Delay.c	123
9.0 Posdelay.c	141
10.0 Saver.c	147
11.0 Data Collection	171
11.1 3com.h.	171
11.2 Data_col.c	173
11.3 S3com.c	225
11.4 Stamp.asm	230
11.5 Netto31.asm	237
12.0 ASAT.c	252
12.1 C3ltoc.c	254
12.2 Internet.asm	260
12.3 F3Ltoc.asm	287
13.0 Data_acq.c	323
14.0 Stripcha.c	340
15.0 Volt3.c	355

List of Figures

Figure 1.1.1 ASAT/PC Network Connections	2
Figure 1.1.2 Intersimulator Network Packet Layout	2
Figure 1.1.3 PC Program Calling Sequence	3

1.0 Introduction

This document is primarily a listing of software developed at IST for the Aviation Technology Research Laboratory, under contract number N61339-89-0045 to PM TRADE. Some chapters cover general purpose libraries, some cover header files included by multiple modules, but most chapters cover the software needed to build one of the executable programs making up the testbed.

The ASAT software delivered to IST consisted of two source files, `asat.c` and `internet.asm`. These underwent minor modifications to allow the delay project to intercept the ASAT packets (see section 1.1 below).

Some of the interface software (e.g., data collection, Section 11) spawns a window process (BE). Being commercial software, IST is not at liberty to supply detailed descriptions of BE. In any case, the intent of its use is apparent from its usage.

1.1 Intersimulator Network Design Description

Intersimulator network service (ISNS) is designed to introduce delays in ASAT simulator communication. Outside of an experimental setting, delays may come from various sources such as satellite latency and computational bottlenecks. As simulations become increasingly complex and geographically disperse delays become a fact of life. ISNS allows evaluation of the impact of packet delays on simulations and simulation users.

The ASAT native communication is through Ethernet, a broadcast LAN. ISNS intercepts ethernet packets, allowing ASAT communication to be controlled for experimentation. ISNS requires a personal computer as well as two ASAT simulators.

ISNS consists of two major components. One part consists of modifications to the ASAT software so only packets correctly addressed are accepted. Further, these modifications cause an ASAT to ignore its own packets. To support this, each ASAT is given a unique identifier, and packets are stamped with the sender's identification.

The second component of ISNS is software which accepts packets, as if it were a LAN, holds the packets for a specified time, alters the packet target addresses, and then re-transmits the packets. Figure 1.1.1 shows the interconnection of the ASATs and the PC running the delay software.

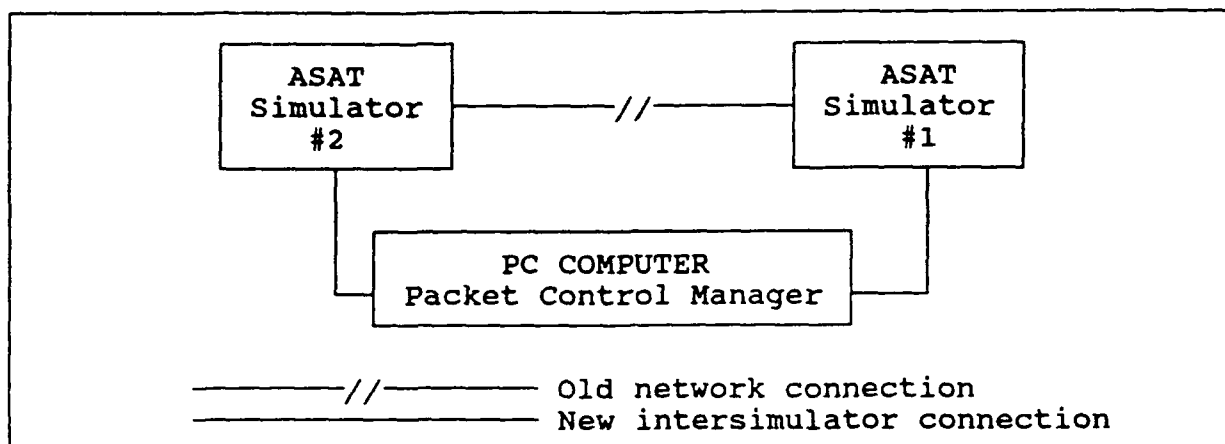


Figure 1.1.1 ASAT/PC Network Connections

Two ASAT source files were modified, *asat.c* and *internet.asm*. The PC software consists of 6 programs: *initsys.exe*, *intersim.exe*, *predelay.exe*, *delay.exe*, *posdelay.exe* and *saver.exe*.

The modifications to the network packets are illustrated in Figure 1.1.2.

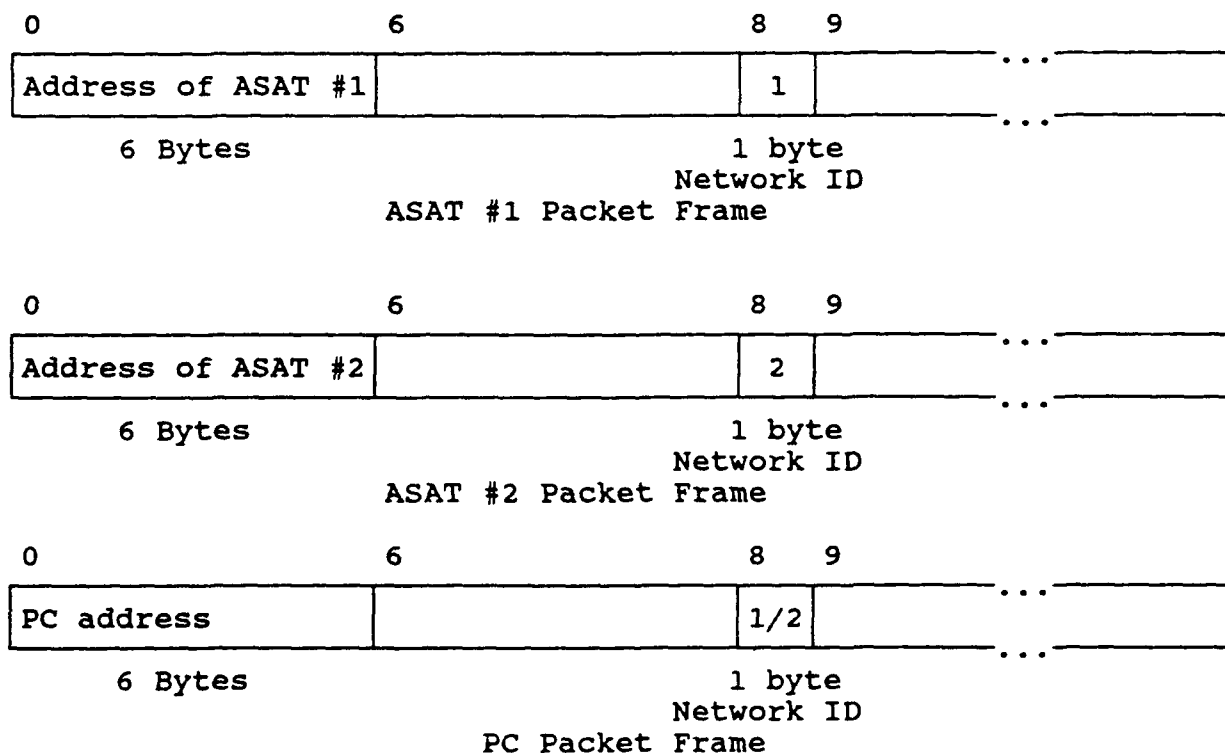


Figure 1.1.2 Intersimulator Network Packet Layout

Initsys.exe carries out ASAT simulator initialization. *Intersim.exe* and *saver.exe* serve as the PC user interface. *Predelay.exe* and *posdelay.exe* support *delay.exe* by carrying out initial and final calculations required by *delay.exe*. *Delay.exe* is, in a real sense, the core of the PC software. It enforces the packet delay. Figure 1.1.3 shows the PC software calling sequences.

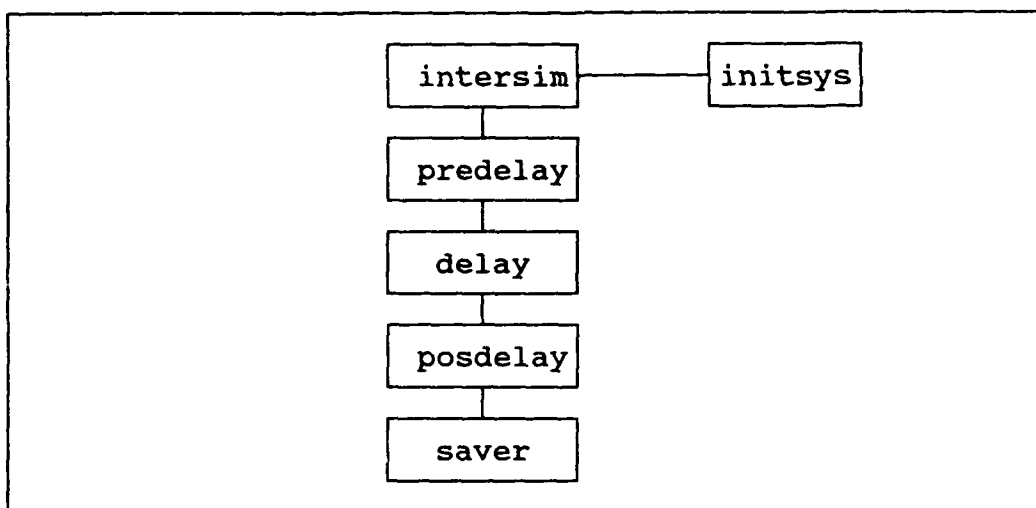


Figure 1.1.3 PC Program Calling Sequence

The program *delay.exe* is constrained to occupy no more than 64K bytes and hence the proliferation of helper functions.

2.0 Software Introduction

The following statement applies to each software element supplied by IST.

```
*****
*           AVIATION   TRAINER   TECHNOLOGY   TESTBED           *
*                                                                 *
*           VERSION   1.0       30 AUGUST 1991                   *
*                                                                 *
*   This software has been developed under the Naval Training    *
*   Systems Center contract N61339-89-C-0045.                    *
*                                                                 *
*   This software is made available for illustrative purposes   *
*   only. It is experimental prototype software which is        *
*   undergoing continual modification, updates and improve-     *
*   ments.                                                        *
*                                                                 *
*   Neither the Institute for Simulation and Training (IST) nor  *
*   the United States Government assumes responsibility for its  *
*   correctness or for any liability resulting directly or       *
*   indirectly from its use.                                     *
*                                                                 *
*   Because this is prototype software, it shall not be proposed *
*   as reusable or as developmental software on subsequent      *
*   Government procurements.                                     *
*                                                                 *
*   IST is unable to provide technical support to users of this  *
*   software; however, IST is interested in receiving comments, *
*   corrections and information relating to the use of any parts *
*   of the software. Comments should be directed to the IST     *
*   with information copy to PM TRADE.                           *
*                                                                 *
*   AVIATION TRAINER TECHNOLOGY LABORATORY                       *
*   Institute for Simulation and Training                         *
*   University of Central Florida                                *
*   12424, Research Parkway, Suite 300                           *
*   Orlando, FL 32826                                           *
*   Phone (407) 658 - 5000                                       *
*                                                                 *
*   Project Manager for Training Devices                         *
*   ATTN: AMCPM-TND-ED                                           *
*   12350 Research Parkway                                       *
*   Orlando, FL 23026-3276                                       *
*   Phone (407) 380 - 8189                                       *
*                                                                 *
*****
```

3.0 System Header Files

The header files covered here are used to build the various programs making up the the IST developed ASAT software. The headers are collected here since they are not, generally, a part of an individual programs.

3.1 Inet.h

The INET header contains various definitions used by the IST ASAT software. It closely parallels *3com.h*, which is included as part of data collection (Section 11). *Inet.h* and *3com.h* have many common elements and some reorganization of these headers may be worthwhile.

```
*****
* inet.h: include file.
*****
#define ADAPTERID      0x0001      /* Ether Link II adapter id */
#define ADAPTERWAIT    0x0060      /* transmitting wait */
#define ASAT1          1           /* ID for simulator #1 */
#define ASAT2          2           /* ID for simulator #2 */
#define ESCKEY         27          /* ESC key code */
#define ASATTYPEBYTE   10          /* Asat packet type byte offset
                                   */
#define ASATSTATEBYTE  51          /* Asat status byte offset in
                                   pkt type 8 */
#define MINLEN         64          /* minimum packet length */
#define MAXLEN         566         /* maximum packet length */
#define MILISEC        0.838*1191.5636 /* time counter reading
                                   converting */
#define ASATERROR1     34.48       /* ave inter-packet time of
                                   Asat1 (ms) */
#define ASATERROR2     71.43       /* ave inter-packet time of
                                   Asat2 (ms) */
#define DEADMARK       0x41        /* simulator spin or explode */
#define NORMALEND      0           /* normal time out */
#define ASAT1DEAD       1           /* Asat #1 is dead */
#define ASAT2DEAD       2           /* Asat #2 is dead */
#define USERSTOP       9           /* User stops the network
                                   service */

#define TEAMMODE        0           /* Team mode - two simulators*/
#define PKTTYPE0        0           /* Asat packet type 0 */
#define PKTTYPE1        1           /* Asat packet type 1 */
#define PKTTYPE8        8           /* Asat packet type 8 */
#define PKTTYPE9        9           /* Asat packet type 9 */
#define PKTTYPE10       10          /* packet type 10 - not Asat
                                   pkt. */

#define TRUE           1
#define FALSE          0

/* DOS driver init request header format */
```

```

typedef struct ini_hdrty
{
char len;                /* header length      */
char non1;
char non2;                /* initial command    */
char non3[2];
char non4[4];
char non5[4];
char non6;                /* number units       */
char cdend[4];            /* code end address    */
char *argo;               /* argument offset     */
short args;               /* argument segment    */
char non7;
}
ini_hdr;

struct WhoStruct           /* Ethernet who-am-I data
                           structure */
{
    unsigned char addr[6]; /* Ethernet adapter address */
    char ver_major;        /* major version */
    char ver_minor;        /* minor version */
    char sub_ver;          /* subversion */
    char type_ds;          /* type version */
    char type_adapter;     /* adapter type */
    char init_status;      /* adapter status */
    char reserved;         /* adapter flags */
    char num_tran_buf;     /* number of xmit buffers */
    short size_tran_buf;   /* xmit buffer size */
    long ttl_tran_cnt;     /* total xmit count */
    long ttl_tran_err_cnt; /* total xmit error count */
    long ttl_tran_timeout_cnt; /* total xmit timeout count */
    long ttl_rcp_cnt;      /* total rcv count */
    long ttl_rcv_bdr_cnt;  /* total broadcast rcv count */
    long ttl_rcv_err_cnt;  /* total rcv error count */
    long ttl_retry_cnt;    /* total xmit retry count */
    char xfr_mode;         /* flag of transfer mode */
    char wait_mode;        /* flag of wait mode */
    char hdr_spec_data;    /* extension pointer */
};

struct PktStr              /* data structure for receiving buffer */
{
    char inp[566];
};

struct HeadStr             /* data structure for packet header */
{
    char inh[36];
};

```

```

struct HeadStr far *Hdptr; /* received packet header pointer */
struct WhoStruct far *Who; /* Who am I structure pointer */
struct PktStr far *Pkt; /* received packet pointer */
struct PktStr far *Pkt1; /* pointer for packet received from
                          ASAT #1 */
struct PktStr far *Pkt2; /* pointer for packet received from
                          ASAT #2 */
struct ini_hdrty *parmsdr; /* parameter structure pointer */

/* Timer variables */

struct dostime_t timein;
unsigned long far *timeptr; /* time counter reading pointer */
unsigned long far *timeQ1; /* time stamp pointer for current
                             pkt */
unsigned long far *timeQ2; /* time stamp pointer for next pkt */

/* transmitting buffer */

char far *Pkttrxptra; /* packet transmitting buffer pointer */
int TotPktsInQ; /* a returned value tells the size of the
                 packet receiving queue */
int network_mode; /* specifies team mode or solitaire mode */

```

3.2 Delay.h

This header is used in building both intersim.exe and saver.exe.

```
*****
* Module       : delay.h                               *
* Programmer   : Sandhya Chandarlapaty                 *
* Purpose      : This header file provides programmer defined *
*                constants needed for the graphic user interface *
*                for the Intersimulator delay study conducted in *
*                the Aviation Trainer Research Laboratory on the *
*                ASAT's.                                  *
* Compilation  : This header file must be included and linked *
*                with the source file, using the Microsoft 6.0 *
*                compiler and linker.                   *
*****
```

```
enum valid_delays {ZERO = 0, SMALL = 250, MEDIUM = 500, LARGE = 750};
```

```
/*
 * defines
 */
```

```
#define BASE 10
#define BACKSPACE 8
#define CARRIAGE 13
#define MAXTRIAL 31
#define BLACK 0
#define BLUE 1
#define GREEN 2
#define CYAN 3
#define RED 4
#define MAGENTA 5
#define BROWN 6
#define WHITE 7
#define DARKGRAY 8
#define LIGHTBLUE 9
#define LIGHTGREEN 10
#define LIGHTCYAN 11
#define LIGHTRED 12
#define LIGHTMAGENTA 13
#define YELLOW 14
#define BRIGHTWHITE 15

#define TRUE 1
#define FALSE 0
```


3.3 Kklib.h

```

/*****
* Module       : kklib.h
* Programmer   : Kevin Kearns
* Purpose      : This header file provides programmer defined
*                constants needed for the screen I/O and keyboard
*                I/O functions provided by the library CLIB.C
* Compilation  : This header file must be included and linked
*                with the source file, using the Turbo C++ 2.0
*                compiler and linker.
*
*****/

#include "conio.h"

/*****
    defines
*****/

#define NullKey      0
#define CtrlD        4
#define Bell         7
#define BackSpaceKey 8
#define TabKey       9
#define CarriageReturn 13
#define CtrlW        23
#define CtrlY        25
#define EscKey       27
#define SpaceKey     32

```

```

#define ShiftTabKey      15+255
#define F1               59+255
#define F2               60+255
#define F3               61+255
#define F4               62+255
#define F5               63+255
#define F6               64+255
#define F7               65+255
#define F8               66+255
#define F9               67+255
#define F10              68+255
#define HomeKey          71+255
#define UpArrow          72+255
#define PgUp             73+255
#define LeftArrow        75+255
#define RightArrow       77+255
#define EndKey           79+255
#define DownArrow        80+255
#define PgDn             81+255
#define InsKey           82+255
#define DelKey           83+255
#define CtrlLeftArrow    115+255
#define CtrlRightArrow   116+255
#define CtrlEnd          117+255
#define CtrlPgDn         118+255
#define CtrlHome         119+255
#define CtrlPgUp         132+255

#define TextKey          32
#define NumberKey        48

#define AltS             31+255
#define AltQ             16+255
#define AltX             45+255

#define False 0
#define True 1
#define OK 1
#define ERROR -1

#define Single_Single 1
#define Double_Double 2
#define Single_Double 3

#define vad(x,y) (((y-1)*160)+((x-1)*2))
#define Min(x,y) (((x) < (y)) ? (x) : (y)) /* returns minimum of
x,y */
#define Max(x,y) (((x) > (y)) ? (x) : (y)) /* returns maximum of
x,y */

#define MAXINPUTLEN      80
#define MAXKEYS          15

```

```
/******  
type definitions  *  
******/  
  
typedef struct  
{  
    int  NUM;  
    int  KEYS[MAXKEYS];  
} KeyList;  
  
typedef unsigned int BYTE;  
typedef unsigned int BOOLEAN;  
  
typedef struct  
{  
    int          *buf;  
    struct text_info  scr_info;  
} screen;
```

4.0 Clib.c

This source code does not directly generate an executable program. Rather, it yields object files which are linked with various components of the testbed software. This library is of a more general nature than the project at hand, and so some components of the library are never exercised by the testbed software.

Program cover sheet

Program name : CLIB.C Latest Version: 1.0 Date: 05/30/91

Languages: C Manufacturer: Borland Version : 2.0

Description:

This file is a library of functions for screen I/O and keyboard I/O. The library can be linked with any C program written in Turbo C and compiled with the Turbo C or Turbo C++ compiler. The C header files *Clib.h* and *Kklib.h* must be included in the source program.

This file consists of helper routines for manipulation of the user interface. Fundamental window capability is also supplied by this module.

```

/*****
 *
 * Source code : Kevin Kearns
 * Documentation: Sandhya Chandarlapaty
 *
 * This file contains some functions
 * necessary for screen I/O and keyboard
 * I/O. It is compiled with the Turbo C++
 * compiler.
 *****/

#include <alloc.h>
#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include <conio.h>
#include <string.h>
#include "kklib.h"
#include "clib.h"

enum (Mono, Color) MonitorType;

int          InsertOn = 0;
int          LastKey;          /* indicates the latest key
                                pressed */

unsigned int  VSeg;
BYTE         HoldCursor;
BYTE         CursorIsOn;      /* indicates if cursor is
                                displayed */

```

```

BYTE          OldTextAttr;          /* stores previous text
                                     attributes */
BYTE          NewTextAttr = (BLUE << 4) | WHITE;
int           WinCount = -1;        /* count of number of
                                     active windows */
WindArray     WindArr[MAXWINDOWS];

/* refer to clib.h for type descriptions of enum {Mono, Color},
   BYTE, WindArray, GetField, GETLIST */

```

```

/*-----
                                GetVideoMode()

PURPOSE   :   sets the value of VSeg and MonitorType
ASSUMES   :   nothing
CALL      :   GetVideoMode();
INPUT PARAMETERS : none
FUNCTION CALLS( other than standard functions ) : none
RETURNS   :   nothing
SIDE EFFECTS : globals Vseg and MonitorType are changed
-----*/

```

```

void GetVideoMode()
{
    struct REGPACK reg;

    reg.r_ax = 0x0F00;
    intr(0x10, &reg);

    if (reg.r_ax == 0x0007)
    {
        MonitorType = Mono;
        VSeg = 0xb000;
    }
    else
    {
        MonitorType = Color;
        VSeg = 0xb800;
    }
}

```

```

/*-----
                CursorOff

PURPOSE   :   sets CursorIsOn to false
ASSUMES   :   nothing
CALL      :   CursorOff();
INPUT PARAMETERS : none
FUNCTION CALLS( other than standard functions ) : none
RETURNS   :   nothing
SIDE EFFECTS : global CursorIsOn is changed
-----*/

```

```

void CursorOff()
{
    union REGS  reg;

    reg.h.ch = 0x20;
    reg.h.ah = 1;
    int86(0x10, &reg, &reg);

    CursorIsOn = False;
}

```

```

/*-----
                                CursorSmall

PURPOSE   :   sets CursorIsOn to true
ASSUMES   :   nothing
CALL      :   CursorSmall();
INPUT PARAMETERS : none
FUNCTION CALLS( other than standard functions ) : none
RETURNS   :   nothing
SIDE EFFECTS : global CursorIsOn is changed
-----*/

```

```

void CursorSmall()
{
    union REGS  reg, oreg;

    switch (MonitorType)
    {
        case Mono :
        {
            reg.h.ch = 0x0C;
            reg.h.cl = 0x0D;
            break;
        }

        case Color :
        {
            reg.h.ch = 6;
            reg.h.cl = 7;
            break;
        }
    }

    reg.h.ah = 1;
    reg.h.bh = 0;
    int86(0x10, &reg, &oreg);
    CursorIsOn = True;
}

```



```

/*-----
                                CursorBig

PURPOSE   :   sets CursorIsOn to true
ASSUMES   :   nothing
CALL      :   CursorBig();
INPUT PARAMETERS : none
FUNCTION CALLS( other than standard functions ) : none
RETURNS   :   nothing
SIDE EFFECTS : global CursorIsOn is changed
-----*/

```

```

void CursorBig()
{
    struct REGPACK reg;

    switch (MonitorType)
    {
        case Mono :
        {
            reg.r_cx = 0x000D;
            break;
        }

        case Color :
        {
            reg.r_cx = 0x0007;
            break;
        }

    }

    reg.r_ax = 0x0100;
    reg.r_bx = 0;
    intr(0x10, &reg);
    CursorIsOn = True;
}

```

```

/*-----
                                SaveCursorStatus()

PURPOSE   :   saves the value of CursorIsOn in HoldCursor
ASSUMES   :   nothing
CALL      :   SaveCursorStatus();
INPUT PARAMETERS : none
FUNCTION CALLS( other than standard functions ) : none
RETURNS   :   nothing
SIDE EFFECTS : global HoldCursor is changed
-----*/

void SaveCursorStatus()
{
    HoldCursor = CursorIsOn;
}

/*-----
                                RestoreCursorStatus()

PURPOSE   :   if CursorIsOn is true, reset it, else it is set to
false
ASSUMES   :   nothing
CALL      :   ResetCursorStatus();
INPUT PARAMETERS : none
FUNCTION CALLS( other than standard functions ) : CursorOff()
RETURNS   :   nothing
SIDE EFFECTS : global CursorIsOn is changed
-----*/

void RestoreCursorStatus()
{
    if (HoldCursor)
        CursorSmall();
    else
        CursorOff();
}

```

```

/*-----
                                SaveScreen

PURPOSE  :  saves cursor status, fill up the tscreen structure with
              the screen information i.e, fill up the scr_info
              structure (which is of type struct text_info) and save
              text in rectangle enclosed by x1,y1, x2,y2 , at
              location pointed to by buf.
ASSUMES   :  nothing
CALL      :  SaveScreen(x1,y1,x2,y2,tscreen);
INPUT PARAMETERS :  int x1, int y1, int x2, int y2, screen *tscreen
                    x1 -- top column number    x2 -- bottom column
                    number
                    y1 -- left line number y2 - right line number
                    tscreen -- ptr to struct of type screen
FUNCTION CALLS (other than standard functions) : SaveCursorStatus()
RETURNS  :  nothing
SIDE EFFECTS :  none
-----*/

void SaveScreen(int x1, int y1, int x2, int y2, screen *tscreen)
{
    SaveCursorStatus();    /* save cursor */

    tscreen->buf = (int *) malloc(((y2-y1) + 1) * ((x2-x1) + 1) *
                                2);
    gettextinfo(&tscreen->scr_info);
    gettext(x1, y1, x2, y2, tscreen->buf);
}

```

```

/*-----
                                RestScreen

PURPOSE   :   copies text from memory on to the screen window, sets
                background and foreground colors, sets the text mode,
                defines an active window for text, and places the
                cursor at the current position.

ASSUMES   :   nothing
CALL      :   RestScreen(x1,y1,x2,y2,tscreen);
INPUT PARAMETERS :   int x1, int y1, int x2, int y2, screen *tscreen
                    x1 -- top column number   x2 -- bottom column
                    number
                    y1 -- left line number y2 - right line number
                    tscreen -- ptr to struct of type screen

FUNCTION CALLS( other than standard functions ):
                    RestoreCursorStatus()

RETURNS   :   nothing
SIDE EFFECTS :   none
-----*/

void RestScreen(int x1, int y1, int x2, int y2, screen *tscreen)
{
    puttext(x1, y1, x2, y2, tscreen->buf); /* put text at x1,y1 */

    free(tscreen->buf);                      /* free up memory */

    textattr(tscreen->scr_info.attribute);
    textmode(tscreen->scr_info.currmode);

                                /* draw window */
    window(tscreen->scr_info.winleft, tscreen->scr_info.wintop,
tscreen->scr_info.winright, tscreen->scr_info.winbottom);
    gotoxy(tscreen->scr_info.curx, tscreen->scr_info.cury);

    RestoreCursorStatus();           /* restore cursor at
                                     current x,y */
}

```

```

/*-----
                                Attr

PURPOSE   :   changes the text attribute, saves previous text
                attribute
ASSUMES   :   nothing
CALL      :   Attr(bg,fg);
INPUT PARAMETERS :   int bg, int fg
                        bg -- background color code
                        fg -- foreground color code
FUNCTION CALLS( other than standard functions ) : none
RETURNS   :   nothing
SIDE EFFECTS :   globals OldTextAttr,NewTextAttr are changed
-----*/

```

```

void Attr(int bg, int fg)
{
    struct text_info  scrn_info;

    gettextinfo(&scrn_info);

    NewTextAttr = (bg << 4) | fg;
    textattr(NewTextAttr);
    OldTextAttr = scrn_info.attribute;
}

```

```

/*-----
                                XYStr

PURPOSE   :   writes s string on the screen at position X,Y
ASSUMES   :   nothing
CALL      :   XYStr(X,Y,s);
INPUT PARAMETERS :   int X, int Y, char s[]
                        X -- column number
                        Y -- line number
                        s -- char string
FUNCTION CALLS( other than standard functions ) : none
RETURNS   :   nothing
SIDE EFFECTS :   none
-----*/

```

```

void XYStr(int x, int y, char s[])
{
    gotoxy(x, y);
    cputs(s);
}

```

```

/*-----
                                PokeChar

PURPOSE   :  puts char ch with attribute att at location x,y on
              screen
ASSUMES   :  nothing
CALL      :  PokeChar(x,y,ch,att);
INPUT PARAMETERS :  int x, int y, int ch, int att
                   x , y -- col, line numbers
                   ch -- char to be put on screen
                   att -- attributes (background and foreground
                           colors) for ch
FUNCTION CALLS( other than standard functions ) : none
RETURNS   :  nothing
SIDE EFFECTS : none
-----*/

```

```

void PokeChar(int x, int y, int ch, int att)
{
    int vch = (ch&255)|(att<<8);
    VPoke(VSeg, vad(x, y), vch);
}

```

```

/*-----
                                VPoke

PURPOSE   :   puts char ch at a given location on screen
ASSUMES   :   nothing
CALL      :   VPoke(vseg, adr, ch);
INPUT PARAMETERS : unsigned vseg, unsigned adr, unsigned ch)
FUNCTION CALLS( other than standard functions ) : none
RETURNS   : nothing
SIDE EFFECTS : none
-----*/

```

```

void VPoke(unsigned vseg, unsigned adr, unsigned ch)
{
    if (vseg == 0xb000)
        poke(vseg, adr, ch);
    else
    {
        _DI = adr;
        _ES = vseg;                                /* color monitor */
        asm cld;
        _BX = ch;
        _DX = 986;

        do
            asm in al,dx;
        while (_AL & 1);

        do
            asm in al,dx;
        while (!(_AL & 1));

        _AL = _BL;

        asm stosb;

        do
            asm in al,dx;
        while (_AL & 1);

        do
            asm in al,dx;
        while (!(_AL & 1));

        _AL = _BH;
        asm stosb;
    }
}

```

```

/*-----
                                PokeStr

PURPOSE   :   puts string pointed to by str, at location x,y on
                screen.  Each char in the string uses NewTextAttr
ASSUMES   :   NewTextAttr has been set
CALL      :   PokeStr(x,y,str);
INPUT PARAMETERS :   int x, int y, char *str
                    x , y -- col, line numbers
                    str -- pointer to char string to be put on
                        screen
FUNCTION   CALLS(   other   than   standard   functions   )   :
PokeChar(x,y,ch,att)
RETURNS   :   nothing
SIDE EFFECTS :   none
-----*/

```

```

void PokeStr(int x, int y, char *str)
{
    int len = strlen(str);
    int i;

    for (i = 0; i < len; i++)
        PokeChar(x+i, y, str[i], NewTextAttr);
}

```

```

/*-----
                                WindowWidth

PURPOSE   :   returns the width of the current window on screen
ASSUMES   :   window has been created
CALL      :   WindowWidth();
INPUT PARAMETERS :   none
FUNCTION CALLS( other than standard functions ) : none
RETURNS   :   returns the width of the window
SIDE EFFECTS :   none
-----*/

```

```

int WindowWidth(void)
{
    struct text_info info;

    gettextinfo(&info);
    return(info.winright - info.winleft);
}

```



```

/*-----
                                CenterText

PURPOSE  : finds the width of the window, and writes a char string
            at the center of the line given by y
ASSUMES  : window has been created
CALL     : CenterText(y,s[]);
INPUT PARAMETERS : int y, char s[]
                    y -- line number on which to write the string
                    s[] -- the char string to be written
FUNCTION CALLS( other than standard functions ) : XYStr(x,y,s)
                                                    WindowWidth()

RETURNS : nothing
SIDE EFFECTS : none
-----*/
void CenterText(int y, char s[])
{
    int x,x1;

    x = WindowWidth() / 2;    /* get window width */
    x1 = strlen(s) / 2;
    XYStr(x-x1, y, s);        /* write string at center of line in
                                window */
}

```

```

/*-----
                                DrawBorder

PURPOSE   :   draws a border for a window
ASSUMES   :   nothing
CALL      :   DrawBorder(x1,y1,x2,y2,border,boxbg,boxfg,title,mess,
                    shadow, explode);
INPUT PARAMETERS :   int x1,y1,x2,y2,border,boxbg,boxfg;char
                    title,mess;
                    int shadow, explode);
                    x1, x2 -- box left and right column numbers
                    y1, y2 -- box top and bottom line numbers
                    border -- type of border
                    Single_Single -- 1
                    Double_Double -- 2
                    Single_double -- 3
                    boxbg, boxfg -- box background and foreground
                                color codes
                    title, mess -- window title, message within
                                window
                    shadow -- 1 if shadow is required for window
                    explode -- 1 if a window needs to be opened and
                                expanded to the box size int x, int
                                y, int att
                    x -- col number
                    y -- line number
                    att -- attributes of char at x,y which are
                                required
FUNCTION CALLS( other than standard functions ) :   PokeStr(x,y,s)
                                                    Attr(bg,fg)
RETURNS :   nothing
SIDE EFFECTS :   none
-----*/

```

```

void DrawBorder(int x1, int y1, int x2, int y2, int border, int
                boxbg, int boxfg, char title[], char mess[], int
                shadow, int explode)
{
    BYTE  Hth;
    BYTE  Len;
    int   x;
    BYTE  contflag;
    BYTE  ratio;
    BYTE  xlmid;
    BYTE  ylmid;
    BYTE  x2mid;
    BYTE  y2mid;
    div_t ans;

    Attr(boxbg, boxfg);
    Len = x2 - x1 + 1;
    Hth = y2 - y1 + 1;

```

```

/*
    if window needs to be expanded slowly, determine ratio, and
    draw window repeatedly ,with a delay of 5 millisecs each
    time, until the required size is obtained.
*/

if (explode)
{
    ans = div(x2-x1, 2);
    x2mid = ans.quot + x1 + 2;
    ans = div(y2-y1, 2);
    y2mid = ans.quot + y1 + 2;
    x1mid = x2mid - 3;
    y1mid = y2mid - 3;

    if ((x2-x1+1) < (y2-1+1))
        ratio = 2;
    else
        ratio = 1;

    contflag = True;

    while (contflag)
    {
        window(x1mid, y1mid, x2mid, y2mid); /* draw a window */
        clrscr(); /* clear the screen */

        if (x1mid >= x1 + (1*ratio)) /* if ximid + ratio
                                     < xi add */
            x1mid = x1mid - (1*ratio); /* ratio to it */
        else
            x1mid = x1; /* else make ximid
                        = xi */

        if (x2mid + (1*ratio) <= x2)
            x2mid = x2mid + (1*ratio);
        else
            x2mid = x2;

        if (y1mid > y1)
            y1mid = y1mid - 1;

        if (y2mid + 1 <= y2)
            y2mid = y2mid + 1;
        else
            y2mid = y2;

        delay(5); /* delay by 5
                  milliseconds */

        if (x1mid == x1 && x2mid == x2 && y1mid == y1 && y2mid
            == y2)
            contflag = False; /* stop when all 4 corners are
                               reached */
    }
}

```

```

window(x1, y1, x2, y2);
clrscr();

/* draw shadow if shadow is required */
if (shadow == 1)
    window(x1, y1, x2+3, y2+1);

/*
    Based on the border value, draw the border
    Single_Single    -- 1
    Double_Double    -- 2
    Single_double    -- 3
*/
switch (border)
{
    case 1 :
    {
        PokeStr(x1, y1, "Z");
        PokeStr(x1+Len-1, y1, "?");
        PokeStr(x1, y1+Hth-1, "@");

        for (x=1; x < Len-1; x++)
        {
            PokeStr(x1+x, y1, "D");
            PokeStr(x1+x, y1+Hth-1, "D");
        }

        for (x=1; x < Hth-1; x++)
        {
            PokeStr(x1, y1+x, "3");
            PokeStr(x1+Len-1, y1+x, "3");
        }

        PokeStr(x2, y2, "Y");
        break;
    }
}

```

```

case 2 :
{
    PokeStr(x1, y1, "I");
    PokeStr(x1+Len-1, y1, ";");
    PokeStr(x1, y1+Hth-1, "H");

    for (x = 1; x < Len-1; x++)
    {
        PokeStr(x1+x, y1, "M");
        PokeStr(x1+x, y1+Hth-1, "M");
    }

    for (x=1; x < Hth-1; x++)
    {
        PokeStr(x1, y1+x, ":");
        PokeStr(x1+Len-1, y1+x, ":");
    }

    PokeStr(x2, y2, "<");
    break;
}

case 3 :
{
    PokeStr(x1, y1, "U");
    PokeStr(x1+Len-1, y1, "8");
    PokeStr(x1, y1+Hth-1, "T");

    for (x=1; x < Len-1; x++)
    {
        PokeStr(x1 + x, y1, "M");
        PokeStr(x1 + x, y1 + Hth - 1, "M");
    }

    for (x = 1; x < Hth-1; x++)
    {
        PokeStr(x1, y1+x, "3");
        PokeStr(x1+ Len-1, y1+x, "3");
    }

    PokeStr(x2, y2, ">");
    break;
}

}

PokeStr(x1+(Len / 2) - (strlen(title) / 2), y1, title); /* put
title */
PokeStr(x1+(Len - (strlen(mess)) - 2), y1+Hth-1, mess); /* put
message */
}

```

```

/*-----
                                DrawShadow

PURPOSE   :   draws a shadow for a window
ASSUMES   :   nothing
CALL      :   DrawShadow(x1,y1,x2,y2);
INPUT PARAMETERS :   int x1,y1,x2,y2
                        x1, x2 -- box left and right column numbers
                        y1, y2 -- box top and bottom line numbers
FUNCTION CALLS( other than standard functions ) : Attr(bg,fg)
PokeChar(x,y ch,att)
RETURNS   :   nothing
SIDE EFFECTS : none
-----*/

```

```

void DrawShadow(int x1, int y1, int x2, int y2)
{
    int Hth;
    int Len;
    int x;
    char buf[4];

    Len = x2 - x1 + 1;
    Hth = y2 - y1 + 1;
    Attr(BLACK, LIGHTGRAY); /* set attributes */

    for (x = 1; x < Hth; x++)
    {
        gettext(x2+1, y1+x, x2+2, y1+x, buf); /* save the text in
                                                window */
        PokeChar(x2+1, y1+x, buf[0], NewTextAttr); /* write back
                                                w/new attr */
        PokeChar(x2+2, y1+x, buf[2], NewTextAttr);

        #if 0
            gotoxy(Len+1, x+1);
            putch(buf[0]);
            putch(buf[2]);
        #endif
    }

    for (x = 2; x < Len+2; x++)
    {
        gettext(x1+x, y1+Hth, x1+x, y1+Hth, buf);
        PokeChar(x1+x, y1+Hth, buf[0], NewTextAttr);
    }
}

```

```

/*-----
                        DrawWindow

PURPOSE   :   draws a window,its border, shadow and sets the new
                attributes for the text
ASSUMES   :   window does not exist
CALL      :   DrawWindow(x1,y1,x2,y2,border,bg,fg,boxbg,boxfg,title,
                mess,shadow, explode);
INPUT PARAMETERS :   int x1,y1,x2,y2,border,boxbg,boxfg;char
                title,mess;int shadow, explode);
                x1, x2 -- box left and right column numbers
                y1, y2 -- box top and bottom line numbers
                border -- type of border
                Single_Single -- 1
                Double_Double -- 2
                Single_double -- 3
                bg, gf -- background and foreground colors for
                        text
                boxbg, boxfg -- box background and foreground
                        color codes
                title, mess -- window title, message within
                        window
                shadow -- 1 if shadow is required for window
                explode -- 1 if a window needs to be opened and
                        expanded to the box size int x, int
                        y, int att
                x -- col number
                y -- line number
                att -- attributes of char at x,y which are
                        required
FUNCTION CALLS(other than standard functions):
                DrawBorder(x1,...explode)
                Drawshadow(x1,y1,x2,y2)
                Attr(bg,fg)
RETURNS   :   nothing
SIDE EFFECTS :   none
-----*/

void DrawWindow(int x1, int y1, int x2, int y2, int border, int bg,
int fg, int boxbg, int boxfg, char title[], char mess[], int
shadow, int explode)
{
    DrawBorder(x1,y1,x2,y2, border, boxbg, boxfg, title, mess,
                shadow, explode);

    if (shadow)
        DrawShadow(x1, y1, x2, y2);

    Attr(bg, fg);

    window(x1+1, y1+1, x2-1, y2-1);

    clrscr();
}

```

```

/*-----
                                SaveWindowParts

PURPOSE   :   Saves the contents, and parameters of the window,
                specified by parameters. These are stored in
                WindArr[++WinCount]. The type description of WindArray
                is in file clib.h

ASSUMES   :   window has been created, and exists

CALL      :   SaveWindowParts(x1,y1,x2,y2,shadow);

INPUT PARAMETERS :   int x1,y1,x2,y2,shadow;
                    x1, x2 -- box left and right column numbers
                    y1, y2 -- box top and bottom line numbers
                    shadow -- 1 if shadow is required for window

FUNCTION CALLS( other than standard functions ) :
                    SaveScreen(x1,...)

RETURNS   :   nothing
SIDE EFFECTS : none
-----*/

```

```

void SaveWindowParts(int x1, int y1, int x2, int y2, int shadow)
{
    if (WinCount < MAXWINDOWS-1)
    {
        if (shadow == 1)
            SaveScreen(x1, y1, x2+3, y2+1,
                       &WindArr[++WinCount].windscr);
        else
            SaveScreen(x1, y1, x2, y2,
                       &WindArr[++WinCount].windscr);

        WindArr[WinCount].x1 = x1;
        WindArr[WinCount].y1 = y1;
        WindArr[WinCount].x2 = x2;
        WindArr[WinCount].y2 = y2;
        WindArr[WinCount].shadow = shadow;
    }
}

```



```

/*-----
                                PushWindow

PURPOSE   :   saves a window, and redraws it, at the same location on
                the screen. This new window is not expanded slowly.
ASSUMES   :   window exists
CALL      :   PushWindow(x1,y1,x2,y2,border,bg,fg,boxbg,boxfg,
                title,mess,shadow);
INPUT PARAMETERS :   int x1,y1,x2,y2,border,boxbg,boxfg;char
                    title,mess;int shadow
                    x1, x2 -- box left and right column numbers
                    y1, y2 -- box top and bottom line numbers
                    border -- type of border
                    Single_Single -- 1
                    Double_Double -- 2
                    Single_double -- 3
                    bg, gf -- background and foreground colors for
                               text
                    boxbg, boxfg -- box background and foreground
                               color codes
                    title, mess -- window title, message within
                               window
                    shadow -- 1 if shadow is required for window
FUNCTION CALLS( other than standard functions ) :
                    SaveWindowParts(x1,...)
                    DrawWindow(x1,.....)

RETURNS   :   nothing
SIDE EFFECTS :   none
-----*/

void PushWindow(int x1, int y1, int x2, int y2, int border, int bg,
                int fg,int boxbg, int boxfg, char title[], char
                mess[], int shadow)
{
    SaveWindowParts(x1, y1, x2, y2, shadow);
    DrawWindow(x1,y1,x2,y2,border,bg,fg,boxbg,boxfg,title,mess,
                shadow,False);
}

```

```

/*-----
ExplodeWindow

PURPOSE  :  saves a window, and redraws it, at the same location on
              the screen. This new window is expanded slowly.
ASSUMES  :  window exists
CALL     :  ExplodeWindow(x1, y1, x2, y2, border, bg, fg, boxbg,
              boxfg, title, mess, shadow);
INPUT PARAMETERS :  int x1,y1,x2,y2,border,boxbg,boxfg;char
                    title,mess;int shadow
                    x1, x2 -- box left and right column numbers
                    y1, y2 -- box top and bottom line numbers
                    border -- type of border
                    Single_Single -- 1
                    Double_Double -- 2
                    Single_double -- 3
                    bg, gf -- background and foreground colors for
                              text
                    boxbg, boxfg -- box background and foreground
                              color codes
                    title, mess -- window title, message within
                              window
                    shadow -- 1 if shadow is required for window
FUNCTION CALLS( other than standard functions ):
                    SaveWindowParts(x1,...)
                    DrawWindow(x1,.....)

RETURNS  :  nothing
SIDE EFFECTS :  none
-----

```

```

void ExplodeWindow(int x1, int y1, int x2, int y2, int border, int
                    bg, int fg, int boxbg, int boxfg, char title[],
                    char mess[], int shadow)
{
    SaveWindowParts(x1, y1, x2, y2, shadow);
    DrawWindow(x1, y1, x2, y2, border, bg, fg, boxbg, boxfg, title,
                mess, shadow, True);
}

```

```

/*-----
                                PopWindow()

PURPOSE   :  pops a window on the screen. copies text from memory on
               to the screen window, sets background and foreground
               colors, sets the text mode, defines an active window for
               text, and places the cursor at the current position,
               reduces the number of saved windows by 1.
ASSUMES   :  window has been previously saved
CALL      :  PopWindow();
INPUT PARAMETERS :  none
FUNCTION CALLS( other than standard functions ) :  RestScreen(....)
RETURNS   :  nothing
SIDE EFFECTS :  none
-----*/

```

```

void PopWindow()
{
    if (WinCount >= 0)
    {
        if (WindArr[WinCount].shadow == 1) /* if window has a
                                             shadow */
            RestScreen( WindArr[WinCount].x1,
                        WindArr[WinCount].y1,
                        WindArr[WinCount].x2+3,
                        WindArr[WinCount].y2+1,
                        &WindArr[WinCount].windscr);
        else
            RestScreen( WindArr[WinCount].x1,
                        WindArr[WinCount].y1,
                        WindArr[WinCount].x2,
                        WindArr[WinCount].y2,
                        &WindArr[WinCount].windscr);

        WinCount--; /* decrement the number of windows */
    }
}

```

```

/*-----
                                DrawBottomBar

PURPOSE  :  writes a character string at the center of the line on
              the screen. If the string has a ` in it, the characters
              following it are highlighted.
ASSUMES  :  nothing
CALL     :  DrawBottomBar(linenum, bstr);
INPUT PARAMETERS :  int linenum; char bstr
                    linenum -- linenum on which to print the
                    character string
                    bstr    -- character string
FUNCTION CALLS( other than standard functions ) :
                    PokeChar(x,y,ch,att)
RETURNS   :  nothing
SIDE EFFECTS :  none
-----*/

```

```

void DrawBottomBar(int linenum, char *bstr)
{
    BOOLEAN onflag = False;
    int x, i;

    x = 1;

    /* put leading blanks */
    for (i = 1; i <= (80-strlen(bstr)) / 2; i++, x++)
        PokeChar(x, linenum, ' ', bNormal);

    /* search for '`' */
    for (i = 0; i < strlen(bstr); i++)
        if (*(bstr+i) == '`')
            onflag = !onflag;
    else
    {
        /* '`' already found */

        if (onflag)
            PokeChar(x, linenum, *(bstr+i), bCharHi);
        else
            PokeChar(x, linenum, *(bstr+i), bNormal);
        x++;
    }

    /* put trailing blanks */
    for (; x < 81; x++)
        PokeChar(x, linenum, ' ', bNormal);
}

```

```

/*-----
                                DrawTopBar

PURPOSE   :   changes the text attributes, saves old attributes,
                writes blank characters on line number 1 with new
                attributes, puts character string at the center of
                line, restores old text attributes.
ASSUMES    :   nothing
CALL       :   DrawTopBar(header);
INPUT PARAMETERS :   char *header
                    header -- pointer to buffer holding the header
                    string
FUNCTION CALLS( other than standard functions ):   Attr(bg,fg)
                                                    PokeChar(x,y,ch,att)
                                                    CenterText(y,s)

RETURNS   :   nothing
SIDE EFFECTS :   none
-----*/

```

```

void DrawTopBar(char *header)
{
    int x;

    Attr(LIGHTGRAY, BLACK); /* set attributes */

    for (x = 1; x < 81; x++)
        PokeChar(x, 1, ' ', NewTextAttr); /* clear line for header
                                           */

    CenterText(1, header);                /* write header */
    textattr(OldTextAttr);
}

```

```

/*-----
                                PosStr

PURPOSE   :   returns the index at which a substring occurs in a
                string
ASSUMES   :   nothing
CALL      :   x = PosStr(str,substr,index);
INPUT PARAMETERS :   char *str, *substr
                    unsigned int index
                    str, substr -- pointers to string and substring
                    index  -- index in string, after which to
                        search for
FUNCTION CALLS( other than standard functions ) : none
RETURNS   :   index where substring is found in string return -1 if not
                found
SIDE EFFECTS : none
-----*/

int PosStr(char* str, char* substr, unsigned int index)
{
    char *ptr;

    ptr = strstr((str+index), substr);

    if (ptr != 0)
        return(ptr - str);
    else
        return -1;
}

```

```

/*-----
                                PosChar

PURPOSE   :   returns the index at which a char occurs in a string,
                after a given index in string
ASSUMES   :   nothing
CALL      :   x = PosChar(str,ch,index);
INPUT PARAMETERS :   char *str, ch
                    int index
                    str -- pointer to string
                    ch -- character to search for
                    index -- index in string, after which to
                        search for
FUNCTION CALLS( other than standard functions ) : none
RETURNS   :   index where character is found in string return -1 if not
                found
SIDE EFFECTS : none
-----*/

int PosChar(char* str, char ch, int index)
{
    int k;
    unsigned int n = strlen(str);

    /* search from index onwards in str */
    for (k = index; ((*str+k) != ch) && (k < n)); k++)
        ;

    if (k == n)
        k = -1;    /* not found after index */

    return k;
}

```

```

/*-----
                                FindLastOccurance

PURPOSE   :   returns the index at which a char occurs in a string,
                starting from the rightmost character in the string
ASSUMES   :   nothing
CALL      :   x = FindLastOccurance(str,ch);
INPUT PARAMETERS :   char *str, ch
                        str -- pointer to string
                        ch -- character to search for
FUNCTION CALLS( other than standard functions ) : none
RETURNS   :   last index where character is found in string
SIDE EFFECTS : the function returns 0 if the ch is not found!!!
-----*/

```

```

int FindLastOccurance(char *str, char ch)
{
    unsigned int n = strlen(str);
    int i = n-1;

    for (; i > 0; i--)
        if (*(str+i) == ch)
            break;

    return(i);
}

```



```

/*-----
                                Insert

PURPOSE   :   inserts a substring in a string, if the given index is
                within the string, else concatenates the substring to
                the string.
ASSUMES    :   nothing
CALL       :   Insert(str,substr,index);
INPUT PARAMETERS :   char *str,*substr,
                    unsigned int index
                    str -- pointer to string
                    substr -- to insert
                    index -- index where to insert
FUNCTION CALLS( other than standard functions ) : none
RETURNS    :   nothing
SIDE EFFECTS : none
-----*/

```

```

void Insert(char* str, char* substr, unsigned int index)
{
    unsigned int m = strlen(substr);
    unsigned int n = strlen(str);

    if (index < n)
    {
        /* index lies within the string */
        memmove((str+index+m), (str+index), n+1-index);
        memmove((str+index), substr, m);
    }
    else
        memmove((str+n), substr, m+1); /* concatenate the substr */
}

```

```

/*-----
                                Delete

PURPOSE   :  deletes a specified number of characters from a string,
               at a specified position in it.
ASSUMES   :  nothing
CALL      :  Delete(str,index,count);
INPUT PARAMETERS :  char *str,*substr,
                   unsigned int index
                   str -- pointer to string
                   count -- number of chars to delete
                   index -- index where to delete
FUNCTION CALLS( other than standard functions ) : none
RETURNS   :  Return -1 if the string is empty or the number of
               characters to be deleted is 0 or if the position in
               the string is invalid. Else return a 0.
SIDE EFFECTS : none
-----*/

```

```

int Delete(char* str, unsigned int index, unsigned int count)
{
    unsigned int n = strlen(str);

    /* if invalid index, set it to 0 */
    if (index < 0)
        index = 0;

    /* if count = 0, return a -1 */
    if (count == 0)
        return -1;

    /* if string is empty, return -1 */
    if (!str[0])
        return -1;

    if (index < n)
    {
        if ((index+count-1) >= n) /* index lies within string */
            str[index] = '\0'; /* exceed string length after
                                deleting */
        else
            memmove((str+index), (str+index+count),
                    n-index-count+1);

        return 0;
    }
    else
        return -1;
}

```

```

/*-----
                                PadLeft

PURPOSE   :   pads a string with leading blanks, until it gets a
                specified length
ASSUMES   :   nothing
CALL      :   PadLeft(str,num);
INPUT PARAMETERS :   (char *str,int num)
                        char *str -- source string
                        int num -- desired length of the string
FUNCTION CALLS( other than standard functions ) :
                        Insert(str,substr,index)
RETURNS   :   Return the string.
SIDE EFFECTS :   none
-----*/

```

```

char *PadLeft(char* str, int num)
{
    while (strlen(str) < num)
        Insert(str, " ", 0);

    return(str);
}

```

```

/*-----
                                PadRight

PURPOSE   :   pads a string with trailing blanks, until it gets a
                specified length
ASSUMES   :   nothing
CALL      :   PadRight(str,num);
INPUT PARAMETERS :   (char *str,int num)
                        char *str -- source string
                        int num -- desired length of the string
FUNCTION CALLS( other than standard functions ) :
                        Insert(str,substr,index)
RETURNS   :   Return the string.
SIDE EFFECTS :   none
-----*/

```

```

char *PadRight(char* str, int num)
{
    while (strlen(str) < num)
        Insert(str, " ", strlen(str)+1);

    return(str);
}

```

```

/*-----
                                LTrim

PURPOSE   :   trims a string on the left, by removing the leftmost
                blanks (or end of string is reached)
ASSUMES   :   nothing
CALL      :   LTrim(str);
INPUT PARAMETERS : char *str -- source string
FUNCTION CALLS( other than standard functions ) :
                Delete(str,index,count)
RETURNS   :   Return the trimmed string.
SIDE EFFECTS : none
-----*/

```

```

char *LTrim(char* str)
{
    int i, count = 0;

    /* traverse string till end/ blank is reached */
    for (i = 0; ((*str+i) != '\0') && ((*str+i) == ' '); i++)
        count++;

    Delete(str, 0, count);

    return(str);
}

```

```

/*-----
                                RTrim

PURPOSE   :   trims a string on the right, by removing the rightmost
                blanks (or end of string is reached)
ASSUMES   :   nothing
CALL      :   RTrim(str);
INPUT PARAMETERS : char *str -- source string
FUNCTION CALLS( other than standard functions ) : none
RETURNS   :   Return the trimmed string.
SIDE EFFECTS : none
-----*/

```

```

char *RTrim(char* str)
{
    int i, count = 0;
    unsigned int len = strlen(str);

    /* traverse string till end/ blank is reached */
    for (i = len-1; ((i >= 0) && ((*str+i) == ' ')); i--)
        count++;

    str[len-count] = '\0';
    return(str);
}

```

```

/*-----
                                Replicate

PURPOSE   :   fills up a string with a given character, upto a given
                length (trims the string till there) and returns it.
ASSUMES   :   string is not empty
CALL      :   Replicate (str,ch,count);
INPUT PARAMETERS : (char *str,char ch, unsigned int count)
                    str -- source string
                    ch  -- char to replace the string characters
                        with
                    count -- index in string till where to replace
FUNCTION CALLS( other than standard functions ) : none
RETURNS    : Return the trimmed string.
SIDE EFFECTS : none
-----*/

```

```

char *Replicate(char *str, char ch, unsigned int count)
{
    memset(str, ch, count);
    str[count] = '\0';
    return(str);
}

```

```

/*-----*/
/*-----*/
/*-----*/
/*-----*/
                                FlushKeybdBuffer

```

```

PURPOSE   :   empties the keyboard buffer
ASSUMES   :   nothing
CALL      :   FlushKeybdBuffer();
INPUT PARAMETERS : none
FUNCTION CALLS( other than standard functions ) : none
RETURNS    : nothing
SIDE EFFECTS : none
-----*/

```

```

void FlushKeybdBuffer()
{
    while (kbhit())
        getch();
}

```

```

/*-----
                                InKey

PURPOSE   :   gets a key hit from keyboard, returns it, sets LastKey
                based on the key hit
ASSUMES   :   nothing
CALL      :   InKey();
INPUT PARAMETERS : none
FUNCTION CALLS( other than standard functions ) : none
RETURNS   :   key hit
SIDE EFFECTS : none
-----*/

```

```

int InKey()
{
    int ch;

    ch = getch();      /* get char from keyboard */

    if (ch == 0)
    {
        ch = getch();
        LastKey = ch+255;
    }
    else
    {
        if (isdigit(ch))
            LastKey = NumberKey;
        else
            if ((ch >= 32) && (ch <= 255))
                LastKey = TextKey;
            else
                LastKey = ch;
    }

    return(ch);
}

```

```

/*-----
                                CheckKeyList

PURPOSE   :   checks if the last key hit is a valid key or not (an
                element in KeyList )
ASSUMES   :   nothing
CALL      :   CheckKeyList(EndKeys);
INPUT PARAMETERS : Keylist EndKeys
FUNCTION CALLS( other than standard functions ) : none
RETURNS   :   boolean
SIDE EFFECTS : none
-----*/
BOOLEAN CheckKeyList(KeyList EndKeys)
{
    int i;
    BOOLEAN ExitFlag;

    i = 1;
    ExitFlag = False;

    /* while no match found with EndKeys element, repeat */
    do
        if (LastKey == EndKeys.KEYS[i])
            ExitFlag = True;                /* found match */
        else
            i++;

    while ((!ExitFlag) && (i <= EndKeys.NUM))
        ;

    return(ExitFlag);
}

```

```

/*-----
                                InputString
PURPOSE   :   gets an input string from the keyboard, checks if the
                last key hit is a valid key or not (an element in
                KeyList ) and if it is takes the appropriate action
                (eg. Backspace key, RightArrow key)
ASSUMES    :   nothing
CALL       :   InputString(s, len, col, row, bg, fg, EndKeys);
INPUT PARAMETERS :   (char *s, int len, int col, int row, int bg,
                        int fg, KeyList EndKeys)
                s - pointer to output string from keyboard
                bg, fg -- background, foreground attributes
                col, row -- position of string
                len -- length of string
                EndKeys -- list of keys other than text
FUNCTION CALLS( other than standard functions ) :
                SaveCursorStatus()
                Attr(bg, fg)
                XYStr(row, col, str)
                CursorSmall()
                CursorBig()
                CheckKeyList(EndKeys)
                RestoreCursorStatus()
RETURNS    :   nothing
SIDE EFFECTS : InsertOn, LastKey get changed
-----*/

```

```

void InputString(char *s, int len, int col, int row, int bg, int
                fg, KeyList EndKeys)
{
    char buff[MAXINPUTLEN];
    int c, cl, bindex, blen;
    BOOLEAN modflag, ClearFlag;

    SaveCursorStatus();    /* save cursor */
    Attr(bg, fg);          /* reset attributes */

    cl = col;
    modflag = False;
    ClearFlag = True;
    bindex = 0;

    memset(buff, ' ', len); /* clear buf */
    buff[len] = '\0';

    XYStr(col, row, buff); /* display buf */
    strcpy(buff, s);
    blen = strlen(s);

    if (blen < len)
        buff[blen] = ' ';
}

```



```

XYStr(col, row, buff);

while (True)
{
    gotoxy(c1, row);

    if (InsertOn)          /* Insert key is pressed */
        CursorBig();
    else
        CursorSmall();

    c = InKey();

    CursorOff();           /* turn cursor off */

    /* take appropriate action, based on key pressed */
    switch (LastKey)
    {
        case HomeKey:
        {
            c1 = col;
            bindex = 0;
            break;
        }

        case RightArrow:
        {
            c1++;
            bindex++;
            ClearFlag = False;
            break;
        }

        case EndKey:
        {
            for (    bindex = len-1;
                    (buff[bindex] == ' ') && (bindex > 0);
                    bindex--)
                ;

            bindex++;
            c1 = col + bindex;
            ClearFlag = False;
            break;
        }
    }
}

```

```

case LeftArrow:
{
    cl--;
    bindex--;
    break;
}

case DelKey:
{
    memmove(buff+bindex, buff+bindex+1, len-bindex-1);
    buff[len-1] = ' ';
    XYStr(cl, row, &buff[bindex]);
    modflag = True;
    ClearFlag = False;
    break;
}

case InsKey:
{
    InsertOn = !InsertOn;
    break;
}

case BackSpaceKey:
{
    if (bindex > 0)
    {
        memmove(buff+bindex-1, buff+bindex,
                len-bindex);
        buff[len-1] = ' ';
        bindex--;
        cl--;
        XYStr(cl, row, &buff[bindex]);
        modflag = True;
    }

    break;
}

```

```

case NumberKey:
case TextKey:
{
    if (InsertOn)
    {
        memmove(buff+bindex+1, buff+bindex,
            len-bindex-1);
        buff[bindex] = c;
        XYStr(c1, row, buff+bindex);
    }
    else
    {
        if (ClearFlag)
        {
            memset(buff, ' ', len);
            buff[len] = '\0';
            XYStr(col, row, buff);
            gotoxy(c1, row);
            ClearFlag = False;
        }

        putchar(c);
        buff[bindex] = c;
    }

    c1++;
    bindex++;
    modflag = True;
    break;
}

if (CheckKeyList(EndKeys))
    break;

if (bindex < 0)
{
    c1 = col;
    bindex = 0;
}
else
if (bindex >= len)
{
    bindex = len - 1;
    c1 = col + bindex;
}
}

```

```

if (modflag)
{
    for (bindex = len-1; (buff[bindex] == ' ') && (bindex > 0);
        bindex--)
        ;

    strncpy(s, buff, bindex+1);
    s[bindex+1] = '\0';
}

RestoreCursorStatus();      /* restore cursor */
}

```

```

/*-----
                                Question

PURPOSE   :   draws a query window on screen, writes a question to
                it, gets the reply, and checks for key pressed, which
                could be ESC or CR. The window is popped off the screen
                and returns the reply.

ASSUMES    :   nothing
CALL       :   Question(question,default_ans,ans,bg,fg);
INPUT PARAMETERS :   (char *question,char *default_ans,char
                        *ans,BYTE bg,BYTE fg)
                        question -- pointer to question string
                        default_ans -- pointer to default_answer string
                        ans -- pointer to answer string
                        bg,fg -- background and foreground colors
FUNCTION CALLS( other than standard functions ) :
                        PushWindow(x1 ..)
                        XYStr(x,y,str)
                        InputString(str,...)
                        PopWindow()
RETURNS    :   the user's reply as a pointer to it
SIDE EFFECTS :   none
-----*/
char *Question(char *question, char *default_ans, char *ans, BYTE
                bg, BYTE fg)
{
    unsigned    x1, x2;
    BOOLEAN     valid_answer;
    KeyList     EndKeys;

    /* draw the question window */
    x1 = 40 - (((strlen(question) + strlen(default_ans)) + 4) / 2);
    x2 = 80 - x1;
    PushWindow(x1,10,x2,14,Single_Single,bg,fg,bg,fg," Question ",
                "", True);
    XYStr(2, 2, question);

    /* set valid key which can be pressed */
    valid_answer = False;
    EndKeys.NUM = 2;
    EndKeys.KEYS[1] = EscKey;
    EndKeys.KEYS[2] = CarriageReturn;

```

```

/* while valid answer not obtained, repeat */
do
{
    strcpy(ans, default_ans);
    InputString(    ans,
                   strlen(default_ans),
                   2+strlen(question)+1,
                   2, BLACK, YELLOW, EndKeys);

    switch (LastKey)
    {
        case CarriageReturn :
        {
            valid_answer = True;
            break;
        }

        case EscKey :
        {
            valid_answer = True;
            strcpy(ans, "N");
            break;
        }
    }
}
while (!valid_answer);

strupr(ans);
PopWindow();    /* pop question window */
return(ans);
}

```

```

/*-----
                                DollarFormat

PURPOSE   :   converts a given string into the dollar format, i.e.
                with a leading '$', and with ',' for every three
                characters starting from the end of the string or from
                the '.' character in it (assumed as decimal point)
ASSUMES   :   nothing
CALL      :   DollarFormat(str);
INPUT PARAMETERS :   (char *str)
                        str -- pointer to character string to be
                        converted
FUNCTION   CALLS(   other than standard functions   )   :
                        PosChar(str,ch,index)
RETURNS   :   pointer to the converted string
SIDE EFFECTS :   none
-----*/

char *DollarFormat(char *str)
{
    unsigned   n = strlen(str);
    int        i = PosChar(str, '.', 0); /* get position of '.' in
str */

    /* if no '.', set it to 4rth position from end */
    if (i == -1)
        i = n-4;
    else
        i-=4;

    /* insert a ',' in str, every fourth position */
    for (; i > 0; i-=4)
        str[i] = ',';

    str[0] = '$'; /* put leading '$' */
    return(str);
}

```

```

/*-----*/
/*-----*/
/*-----*/

```

InitGet

```

PURPOSE : a variable of structure GetHead is initialized
ASSUMES : nothing
CALL : InitGet(g);
INPUT PARAMETERS : (GetHead *g)
                  g -- pointer to structure of type GetHead
FUNCTION CALLS( other than standard functions ) : none
RETURNS : nothing
SIDE EFFECTS : none
-----*/

```

```

void InitGet(struct GetHead *g)
{
    g->First = NULL;
    g->Last = NULL;
    g->EndKeys.NUM = 3;
    g->EndKeys.KEYS[1] = EscKey;
    g->EndKeys.KEYS[2] = PgDn;
    g->EndKeys.KEYS[3] = PgUp;
    return;
}

```



```

/*-----
                                AddGet

PURPOSE   :   adds a new node of structure GetField, into the linked
                list pointed by g (which is a structure GetHead). All
                the fields in the new node are appropriately
                initialized
ASSUMES   :   g is initialized by InitGet
CALL      :   AddGet(g,x,y,len,saystr,getstr,format,help,ignore);
INPUT PARAMETERS : (struct GetHead *g, BYTE x, BYTE y, BYTE len,
                    char *saystr, char *getstr, InputType format,
                    BYTE help, BOOLEAN ignore)
FUNCTION   CALLS(   other than standard functions   )   :
                    Replicate(str,ch,count)
RETURNS   :   nothing
SIDE EFFECTS : none
-----*/

```

```

void AddGet(struct GetHead *g, BYTE x, BYTE y, BYTE len, char
*saystr, char *getstr, InputType format, BYTE help, BOOLEAN ignore)
{
    struct GetField *newg;

    /* allocate mem for new node */
    newg = (struct GetField *) malloc(sizeof(struct GetField));

    newg->Next = NULL;

    if (g->First == NULL)
    {
        /* if this is the first node in the linked list */
        g->First = newg;
        g->Last = newg;
        newg->Prev = NULL;
    }
    else
    {
        /*add it after the last node */
        g->Last->Next = newg;
        newg->Prev = g->Last;
        g->Last = newg;
    }
}

```

```

/* set feild values in new node */
newg->X = x;
newg->Y = y;
newg->Len = len;
newg->Help = help;
newg->Ignore = ignore;
newg->Format = format;
newg->Validation = NULL;
newg->Execute = NULL;
newg->Buff = getstr;
newg->WorkStr = (char *) malloc(len+1);
strcpy(newg->WorkStr, '\0');
newg->SayStr = (char *) malloc(strlen(saystr)+1);
strcpy(newg->SayStr, saystr);
newg->DummyStr = (char *) malloc(len+1);
strcpy(newg->DummyStr, '\0');
newg->Mask = (char *) malloc(len+1);
strcpy(newg->Mask, '\0');

/* set the Mask field based on the format value */
switch (format)
{
    case AlphaNumeric:
    case Numeric:
    {
        newg->Mask = (char *) Replicate(newg->Mask, '_', len);
        break;
    }

    case Numeric_Decimal:
    {
        newg->Mask = (char *) Replicate(newg->Mask, '_', len);
        newg->Mask[len-3] = '.';
        break;
    }

    case Percent_Decimal:
    {
        newg->Mask = (char *) Replicate(newg->Mask, '_', len);
        newg->Mask[len-1] = '%';
        newg->Mask[len-3] = '.';
        break;
    }
}

```

```

case Percent:
{
    newg->Mask = (char *) Replicate(newg->Mask, '_', len);
    newg->Mask[len-1] = '%';
    break;
}

case Dollar:
{
    newg->Mask = (char *) Replicate(newg->Mask, '_', len);
    DollarFormat(newg->Mask);
    break;
}

case Dollar_Cents:
{
    newg->Mask = (char *) Replicate(newg->Mask, '_', len);
    newg->Mask[len-3] = '.';
    DollarFormat(newg->Mask);
    break;
}

case Date:
{
    strcpy(newg->Mask, "__/__/__\0");
    break;
}

case TimeCode:
{
    newg->Mask = (char*) Replicate(newg->Mask, '_', len);
    newg->Mask[2] = ':';
    newg->Mask[5] = ':';
    newg->Mask[8] = ':';
    break;
}
}
}

```

```

/*-----
                                FormatString

PURPOSE   :   changes the contents of the field DummyStr in the
                structure GetField, based on the field Format in it
ASSUMES   :   input structure has been initialized
CALL      :   FormatString(gptr);
INPUT PARAMETERS : (struct GetField *gptr)
                    struct GetField *gptr -- pointer to structure
                    GetField
FUNCTION CALLS( other than standard functions ) :
                    PosStr(str,ch,pos)
                    PadRight(str,count)

RETURNS   :   nothing
SIDE EFFECTS : none
-----*/

```

```

void FormatString(struct GetField *gptr)
(
    int    i;
    int    buflen;
    int    buffdecpos;
    int    maskdecpos;
    char *tempstr;

    /* copy Mask into DummyStr */

    strcpy(gptr->DummyStr, gptr->Mask);

    /* replace '-' in DummyStr by ' ' */

    for (i=0; i < strlen(gptr->DummyStr); i++)
        if (gptr->DummyStr[i] == '-')
            gptr->DummyStr[i] = ' ';

    buffdecpos = PosChar(gptr->Buff, '.', 0); /* get index of '.'
                                                in Buff*/
    buflen = strlen(gptr->Buff)-1;
    maskdecpos = PosChar(gptr->DummyStr, '.', 0);

    if (maskdecpos == -1)
        maskdecpos = strlen(gptr->DummyStr)-1;

```

```

/* if Format is AlphaNumeric, copy Buff into DummyStr, from
maskdecpos*/

if ((buffdecpos != -1) && (gptr->Format != AlphaNumeric))
{
    maskdecpos = PosChar(gptr->DummyStr, '.', 0) + 1;

    for (i = buffdecpos+1; i <= buflen; i++)
    {
        gptr->DummyStr[maskdecpos] = gptr->Buff[i];
        maskdecpos++;
    }

    buflen = buffdecpos - 1;
    maskdecpos = PosChar(gptr->DummyStr, '.', 0) - 1;
}

/*if Format is AlphaNumeric or Date, copy Buff into tempStr,
pad trailing blanks, copy it back into DummyStr*/

if (gptr->Format == AlphaNumeric || gptr->Format == Date)
{
    tempstr = (char *) malloc(gptr->Len + 1);
    strcpy(tempstr, gptr->Buff);
    tempstr = (char *)
        PadRight(tempstr, strlen(gptr->DummyStr) -
            strlen(gptr->Buff));

    strcpy(gptr->DummyStr, tempstr);
    free(tempstr);

    if (gptr->Format == Date)
    {
        gptr->DummyStr[2] = '/';
        gptr->DummyStr[5] = '/';
    }
}
else
for (i = buflen; i >= 0; i--)
{
    if (gptr->DummyStr[maskdecpos] != ' ')
        do
            maskdecpos--;
        while (gptr->DummyStr[maskdecpos] != ' ');

    gptr->DummyStr[maskdecpos] = gptr->Buff[i];
    maskdecpos--;
}

return;
}

```

```

/*-----
                                DisplayGets

PURPOSE   :   displays the contents of the field SayStr,DummyStr in
                the structure GetField, based on the field Format in
                it. This is done for all such nodes in the linked list
                pointed by the head g (struct GetHead)
ASSUMES   :   input structure has been initialized
CALL      :   DisplayGets(g);
INPUT PARAMETERS : (struct GetHead *g)
                    struct GetHead *g -- pointer to structure
                    GetHead
FUNCTION CALLS( other than standard functions ) :
                    Attr(bg,fg)
                    XYStr(x,y,str)
                    PadRight(str,count)
                    FormatString(gptr)

RETURNS   :   nothing
SIDE EFFECTS : none
-----*/

void DisplayGets(struct GetHead g)
{
    struct GetField  *lcptr;
    int              saylen;

    lcptr = g.First;          /* start at first node */

    /* repeat while last node is not reached */
    while (lcptr != NULL)
    {
        Attr(BLUE, WHITE);    /* set attributes */
        XYStr(lcptr->X, lcptr->Y, lcptr->SayStr);    /* display
                                                    SayStr */
        saylen = strlen(lcptr->SayStr)+1;

        if (lcptr->Ignore) /* if Ignore is on, change attributes */
            Attr(BLACK, LIGHTGRAY);
        else
            Attr(BLACK, YELLOW);

        strcpy(lcptr->DummyStr, lcptr->Buff);    /* copy Buff into
                                                    DummyStr */

        /* if Format is AlphaNumeric,
           pad DummyStr with trailing blanks display it */
    }
}

```

```

switch (lcptr->Format)
{
    case AlphaNumeric:
    {
        XYStr(lcptr->X + saylen, lcptr->Y,
        PadRight(lcptr->DummyStr, lcptr->Len));
        break;
    }

    default:
    {
        FormatString(lcptr);
        XYStr(lcptr->X + saylen, lcptr->Y,
        PadRight(lcptr->DummyStr, lcptr->Len));
        break;
    }
}

lcptr = lcptr->Next;      /* go to the next node in linked
                           list */
}

return;
}

/*-----
                                ClearLine

PURPOSE  :  copies Mask field into DummyStr field, replaces all '-'
            in it by ' ', displays it and copies in into WorkStr
            field
ASSUMES   :  input structure has been initialized
CALL      :  ClearLine(lcptr);
INPUT PARAMETERS :  (struct GetField *lcptr)
                   struct GetField *lcptr -- pointer to structure
                   GetField
FUNCTION CALLS( other than standard functions ) :
                   XYStr(x,y,str)
RETURNS  :  nothing
SIDE EFFECTS :  none
-----*/
void ClearLine(struct GetField *lcptr)
{
    int i;

    /* copy Mask into DummyStr */
    strcpy(lcptr->DummyStr, lcptr->Mask);

    /* replace '-' by ' ' */
    for (i = 0; i < strlen(lcptr->DummyStr); i++)
        if (lcptr->DummyStr[i] == '-')
            lcptr->DummyStr[i] = ' ';
}

```

```
/* display it */  
    XYStr(lcptr->X+strlen(lcptr->SayStr)+1,    lcptr->Y,  
lcptr->DummyStr);  
/* copy DummyStr into WorkStr */  
strcpy(lcptr->WorkStr, lcptr->DummyStr);  
return;  
}
```



```

/*-----*/
/*-----DeleteChar
PURPOSE : deletes a specified number of characters from a string,
          at a specified position in it, based on the value of
          form(InputType).
ASSUMES : nothing
CALL    : DeleteChar(char *buff, int p, BYTE len, InputType
          form);
INPUT PARAMETERS : char *buff, int p, BYTE len, InputType form
                  buff -- pointer to string
                  p -- index where to delete
                  len - how much to delete
FUNCTION CALLS( other than standard functions ) :
                  Delete(str,count, index)
                  PosChar(str,char,index)
                  Insert(str,substr,index)
RETURNS : nothing
SIDE EFFECTS : none
/*-----*/
void DeleteChar(char *buff, int p, BYTE len, InputType form)
{
    int decpos;

    if (p < 0) /* invalid index, do nothing */
        return;

    /* based on the value of form, take the appropriate action */
    switch (form)
    {
        case Percent:
        {
            Delete(buff, p, 1);
            Insert(buff, " ", len-2);
            break;
        }

        case Dollar_Cents:
        {
            if (p == 0)
                return;
        }
    }
}

```

```

case Numeric_Decimal:
case Percent_Decimal:
{
    Delete(buff, p, 1);
    decpos = PosChar(buff, '.', 0);

    if (p <= decpos)
    {
        if (form == Dollar_Cents)
            Insert(buff, "-", 1);
        else
            Insert(buff, " ", 0);
    }

    if (p > decpos)
    {
        if (form == Percent_Decimal)
            Insert(buff, "%", len-2);
        else
            Insert(buff, " ", len-1);
    }

    break;
}

case Date:
{
    Delete(buff, p, 1);

    if ((p == 1) || (p == 4) || (p == 7))
        Insert(buff, " ", p);
    else
        Insert(buff, " ", p+1);
    break;
}

default:
{
    Delete(buff, p, 1);
    strcat(buff, " ");
    break;
}
}

return;
}

```

```

/*-----
                                AddChar

PURPOSE  :  deletes a specified number of characters from a string,
              at a specified position in it, based on the value of
              form(InputType) and inserts a given char in it after
              the deletion
ASSUMES   :  nothing
CALL      :  AddChar(char *buff, BYTE ch,int p, InputType form);
INPUT PARAMETERS : char *buff, int p, BYTE ch, InputType form
                  buff -- pointer to string
                  p -- index where to delete
                  ch -- character to insert
FUNCTION CALLS( other than standard functions ) :
                  Delete(str,count, index)
                  PosChar(str,char,index)
                  Insert(str,substr,index)
RETURNS   :   False if form is not AlphaNumeric or if ch is not a
                  digit or is not a '.', else return True
SIDE EFFECTS : none
-----*/

```

```

BOOLEAN AddChar(char *buff, BYTE ch, int p, InputType form)
(
    int  decpos;
    char *dummy;

    /* form is not AlphaNumeric/ ch not a digit nor a '.' */
    if ((form != AlphaNumeric) && (((ch < 48) || (ch > 57)) && (ch
!= 46)))
        return(False);

    if (!InsertOn)          /* Insert key is released? */
        buff[p] = ch;      // Flipped else/if
    else
    {
        if (p > strlen(buff)-1)
            buff[p] = ch;
        else
        {
            dummy = (char *) malloc(2);
            dummy[0] = ch;
            dummy[1] = '\0';
            decpos = PosChar(buff, '.', 0);

```

```

if (decpos == -1)
{
    if (form == Date)
    {
        switch (p)
        {
            case 0:
            case 1:
            {
                Delete(buff, 1, 1);
                Insert(buff, dummy, 1);
                break;
            }

            case 3:
            case 4:
            {
                Delete(buff, 4, 1);
                Insert(buff, dummy, 4);
                break;
            }

            case 6:
            case 7:
            {
                Delete(buff, 7, 1);
                Insert(buff, dummy, 7);
                break;
            }
        }
    }
    else
    {
        Delete(buff, strlen(buff)-1, 1);
        Insert(buff, dummy, p);
    }
}
else

```

```

    {
        if (p > decpos)
        {
            if ((form == Percent) || (form ==
                Percent_Decimal))
            {
                Delete(buff, strlen(buff)-2, 1);
                Insert(buff, dummy, p);
            }
            else
            {
                Delete(buff, strlen(buff)-1, 1);
                Insert(buff, dummy, p);
            }
        }
        else
        {
            Delete(buff, decpos-1, 1);
            Insert(buff, dummy, p);
        }
    }

    free(dummy);
}

return(True);
}

```

```

/*-----
                                     TextChar

PURPOSE   :   deletes a specified number of characters from the
                WorkStr field and Inserts DummyStr into it based on
                Format field, and whether value at input variable p is
                greater than or less than index of decimal point in the
                WorkStr.
ASSUMES   :   nothing
CALL      :   TextChar(*lcptr, ch, *p, *ClearFlag)
INPUT PARAMETERS : (struct GetField *lcptr, int ch, int *p, BOOLEAN
                    *ClearFlag)
                                *lcptr -- pointer to struct GetField
                                *p -- pointer to index where to delete
                                ch -- character to insert
FUNCTION CALLS( other than standard functions ) :
                                ClearLine(lcptr)
                                Delete(str,count, index)
                                PosChar(str,char,index)
                                Insert(str,substr,index)
                                Replicate(str,ch,count)

RETURNS   :   nothing
SIDE EFFECTS : none
-----*/

```

```

void TextChar(struct GetField *lcptr, int ch, int *p, BOOLEAN
*ClearFlag)
{
    int decpos;

    if (*ClearFlag)                /* ClearFlag is on */
    {
        ClearLine(lcptr);
        *ClearFlag = False;
    }

    if (ch == '.')
    {
        if ((lcptr->Format == Percent_Decimal) ||
            (lcptr->Format == Dollar_Cents) ||
            (lcptr->Format == Numeric_Decimal))
        {
            /* Format could be
               Percent_decimal,
               Dollar_cents,
               Numeric_Decimal */

            decpos = PosChar(lcptr->Mask, '.', 0); /* get position
                                                       of '.' */

```

```

        if (*p < decpos)
        {
            Delete(lcptr->WorkStr, *p, decpos-*p);

            if (lcptr->Format == Dollar_Cents)
                Insert( lcptr->WorkStr,
                        Replicate(lcptr->DummyStr,
                                ' ', decpos - *p), 1);
            else
                Insert( lcptr->WorkStr,
                        Replicate(lcptr->DummyStr,
                                ' ', decpos - *p), 0);
        }

        *p = decpos + 1;
        return;
    }
    else
        if (lcptr->Format != AlphaNumeric)
            return;
    }

    if (AddChar(lcptr->WorkStr, ch, *p, lcptr->Format))
    {
        do
            (*p)++;
        while ((lcptr->Mask[*p] != '_') && (*p < lcptr->Len));

        if (*p < lcptr->Len)
            return;

        do
            (*p)--;
        while ((lcptr->Mask[*p] != '_') && (*p > -1));
    }
}

```

```

/*-----
                                BackSpaceChar
PURPOSE   :   deletes a specified number of characters from the
                WorkStr field and Inserts DummyStr into it based on
                Format field, and whether value at input variable p is
                greater than or less than index of decimal point in the
                WorkStr.
ASSUMES   :   nothing
CALL      :   BackSpaceChar(*lcptr, ch, *p, *ClearFlag)
INPUT PARAMETERS : (struct GetField *lcptr, int *p, BOOLEAN
*ClearFlag)
                *lcptr -- pointer to struct GetField
                *p -- pointer to index where to delete
FUNCTION CALLS( other than standard functions ) :
                Delete(str,count, index)
                PosChar(str,char,index)
                Insert(str,substr,index)
                Replicate(str,ch,count)
RETURNS   :   nothing
SIDE EFFECTS : none
-----*/

void BackSpaceChar(struct GetField *lcptr, int *p, BOOLEAN
*ClearFlag)
{
    int decpos;

    if ((lcptr->Format == Percent_Decimal) ||
        (lcptr->Format == Dollar_Cents) ||
        (lcptr->Format == Numeric_Decimal))
    {
        /* Format could be Percent_decimal, Dollar_cents,
        Numeric_Decimal */
        decpos = PosChar(lcptr->WorkStr, '.', 0);

        if (*p < decpos)
        {
            *ClearFlag = False;
            DeleteChar(lcptr->WorkStr, (*p)-1, lcptr->Len,
lcptr->Format);
        }
        else
        {
            do
            {
                (*p)--;
                while ((lcptr->Mask[*p] != '_') && (*p < lcptr->Len));

                if (*p != -1)
                {
                    *ClearFlag = False;
                    DeleteChar(lcptr->WorkStr, *p, lcptr->Len,
lcptr->Format);
                }
            }
            while (*p > 0);
        }
    }
    else

```



```

if (*p != 0)
{
    do
        (*p)--;
        while ((lcptr->Mask[*p] != '_') && (*p < lcptr->Len));

        if (*p != -1)
        {
            *ClearFlag = False;
            DeleteChar(lcptr->WorkStr, *p, lcptr->Len,
                lcptr->Format);
        }
    }
    return;
}

```

```

/*-----
CleanNumberForDisplay
PURPOSE : cleans up an a given string removing unwanted
          characters, based on the value of the given format. The
          string is made ready for display.
ASSUMES : nothing
CALL : CleanNumberForDisplay(*buff, form)
INPUT PARAMETERS : (char *buff, InputType form)
                   *buff -- ptr to string to be cleaned
                   form -- based on which buff is to be cleaned
FUNCTION CALLS( other than standard functions ) :
               Delete(str,count, index)
               LTrim(str)
               RTrim(str)
               Insert(str,substr,index)
RETURNS : nothing
SIDE EFFECTS : none
-----*/

```

```

void CleanNumberForDisplay(char *buff, InputType form)
{
    BYTE len;
    int i;
    char dummy[3];

    if (form == Date)
    {
        /* form is Date */
        /* copy first two chars from buff */
        strncpy(&dummy[0], buff, 2);
        dummy[2] = '\0';

        /* trim dummy from both sides; if only one char left, add
           a leading 0 */
        /* do this for month, day, year and return */

        if (strlen(LTrim(RTrim(dummy))) == 1)
        {
            buff[0] = '0';
            buff[1] = dummy[0];
        }

        strncpy(&dummy[0], buff+3, 2);
        dummy[2] = '\0';

        if (strlen(LTrim(RTrim(dummy))) == 1)
        {
            buff[3] = '0';
            buff[4] = dummy[0];
        }
    }
}

```

```

    strncpy(&dummy[0], buff+6, 2);
    dummy[2] = '\0';

    if (strlen(LTrim(RTrim(dummy))) == 1)
    {
        buff[6] = '0';
        buff[7] = dummy[0];
    }

    return;
}

/* form is not Date */

i = 0;
len = strlen(buff);

/* search for an integer */

while (!isdigit(buff[i]) && i < len)
    i++;

i++;
/* search for '.' */
while ((i < len) && (buff[i] != '.'))
{
    if (buff[i] == ' ')
    {
        /* if char is a ' ', remove it */
        Delete(buff, i, 1);

        /* if form is Dollar or Dollar_cents, insert a blank at
           pos 1 */
        if ((form == Dollar) || (form == Dollar_Cents))
            Insert(buff, " ", 1);
        else
            Insert(buff, " ", 0); /* else insert blank at pos
                                   0 */
    }
    i++;
}

return;
}

```

```

/*-----
                                CleanEntry

PURPOSE   :   cleans the WorkStr field of the input parameter based
                on the Format type. The leading and trailing blanks are
                removed, leading zeros are removed and so on.

ASSUMES   :   nothing
CALL      :   Cleanentry(*lcptr)
INPUT PARAMETERS : (struct GetField *lcptr)
                *lcptr -- pointer to struct GetField
FUNCTION CALLS( other than standard functions ) :
                Delete(str,count, index)
                RTrim(str)
                LTrim(str)
RETURNS   :   nothing
SIDE EFFECTS : none
-----*/

```

```

void CleanEntry(struct GetField *lcptr)
{
    /* if WorkStr is empty, return */
    if (strcmp(lcptr->WorkStr, "") == 0)
        return;

    /* based on Format , clean it */
    switch (lcptr->Format)
    {
        case Numeric:
        case AlphaNumeric:
        {
            lcptr->WorkStr = (char *) RTrim(LTrim(lcptr->WorkStr));
            break;
        }

        case Dollar:
        case Dollar_Cents:
        {
            Delete(lcptr->WorkStr, 0, 1);
            lcptr->WorkStr = (char *) RTrim(lcptr->WorkStr);

            if (lcptr->WorkStr[strlen(lcptr->WorkStr)-1] == '.')
                Delete(lcptr->WorkStr, strlen(lcptr->WorkStr)-1, 1);

            lcptr->WorkStr = (char *) LTrim(lcptr->WorkStr);
            break;
        }
    }
}

```

```

    case Numeric_Decimal:
    {
        lcptr->WorkStr = (char *) RTrim(lcptr->WorkStr);

        if (lcptr->WorkStr[strlen(lcptr->WorkStr)] == '.')
            Delete(lcptr->WorkStr, strlen(lcptr->WorkStr)-1, 1);

        lcptr->WorkStr = (char *) LTrim(lcptr->WorkStr);
        break;
    }

    case Percent:
    case Percent_Decimal:
    {
        Delete(lcptr->WorkStr, strlen(lcptr->WorkStr)-1, 1);
        lcptr->WorkStr = (char *) RTrim(lcptr->WorkStr);

        if (lcptr->WorkStr[strlen(lcptr->WorkStr)-1] == '.')
            Delete(lcptr->WorkStr, strlen(lcptr->WorkStr)-1, 1);

        lcptr->WorkStr = (char *) LTrim(lcptr->WorkStr);
        break;
    }
}

strcpy(lcptr->Buff, lcptr->WorkStr);
return;
}

```

```

/*-----
                                UpDateLcptr

```

```

PURPOSE   :  updates a node in the list pointed to by Lcptr, based
               on value of LastKey
ASSUMES   :  nothing
CALL      :  UpDateLcptr(*lcptr)
INPUT PARAMETERS : (struct GetField *lcptr)
                  *lcptr -- pointer to struct GetField
FUNCTION CALLS( other than standard functions ) :none
RETURNS   :  nothing
SIDE EFFECTS : none

```

```

-----*/
struct GetField *UpdateLcptr(struct GetField *lcptr)
{
    BOOLEAN flag;

    flag = False;

```

```

while (!flag)
{
    if ((LastKey == UpArrow) || (LastKey == ShiftTabKey))
    {
        do
        {
            if (lcptr->Prev != NULL)
            {
                lcptr = lcptr->Prev;
                if (!lcptr->Ignore)
                    flag = True;
            }
            else
                if (!lcptr->Ignore)
                    flag = True;

            LastKey = NullKey;
        } while ((lcptr->Prev != NULL) && (!flag));
    }
    else
    {
        do
        {
            if (lcptr->Next != NULL)
            {
                lcptr = lcptr->Next;

                if (!lcptr->Ignore)
                    return(lcptr);
            }
            else
                if (LastKey == CarriageReturn)
                    return(NULL);
            else
                if ((lcptr->Next == NULL) && (lcptr->Ignore))
                    LastKey = UpArrow;
            else
                return(lcptr);
        } while ((lcptr->Next != NULL) && (!flag));
    }
}
return(lcptr);
}

```

```

/*-----
                                Get

ASSUMES   :  nothing
CALL      :  Get(g)
INPUT PARAMETERS :  (struct GetHead g)
                        g -- variable of type struct GetField
FUNCTION CALLS( other than standard functions ) :
                        SaveCursorStatus
                        DisplayGets(g)
                        CursorSmall()
                        CursorBig()
                        CursorOff()
                        UpDateLcptr(lcptr)
                        Attr(bg,fg)
                        FormatString(lcptr)
                        XYStr(str,x,y)
                        CleanEntry(lcptr)
                        RestoreCursorStatus()

RETURNS   :  nothing
SIDE EFFECTS :  none
-----*/

```

```

void Get(struct GetHead g)
{
    BOOLEAN EndEditFlag;
    BOOLEAN ExitGetFlag;
    BOOLEAN ClearFlag;
    struct GetField *lcptr;
    int saylen, p;
    BYTE ch;

    SaveCursorStatus();
    DisplayGets(g);
    lcptr = g.First;
    CursorSmall();
    ExitGetFlag = False;
    LastKey = DownArrow;

    if (lcptr->Ignore)
        lcptr = UpdateLcptr(lcptr);
}

```

```

do
{
    Attr(BLACK, YELLOW);

    if (strcmp(lcptr->WorkStr, "\0") == 0)
    {
        strcpy(lcptr->WorkStr, lcptr->Buff);
        strcpy(lcptr->DummyStr, lcptr->Mask);
        FormatString(lcptr);
        strcpy(lcptr->WorkStr, lcptr->DummyStr);
    }

    p = 0;

    while (lcptr->Mask[p] != '_')
        p++;

    EndEditFlag = False;
    ClearFlag = True;
    saylen = strlen(lcptr->SayStr) + 1;

    do
    {
        Attr(BLACK, YELLOW);
        XYStr(lcptr->X + saylen, lcptr->Y, lcptr->WorkStr);
        gotoxy(lcptr->X + saylen + p, lcptr->Y);

        if (InsertOn)
            CursorBig();
        else
            CursorSmall();

        ch = InKey();
        CursorOff();

        switch (LastKey)
        {
            case TextKey:
            case NumberKey:
            {
                TextChar(lcptr, ch, &p, &ClearFlag);
                break;
            }

            case InsKey:
            {
                ClearFlag = False;
                InsertOn = !InsertOn;
                break;
            }
        }
    }

```



```

case BackSpaceKey:
{
    BackSpaceChar(lcptr, &p, &ClearFlag);
    break;
}

case DelKey:
{
    ClearFlag = False;
    DeleteChar(lcptr->WorkStr, p, lcptr->Len,
               lcptr->Format);
    break;
}

case RightArrow:
{
    ClearFlag = False;

    do
        p++;
    while ((lcptr->Mask[p] != '_') && (p <
        lcptr->Len));

    if (p < lcptr->Len)
        break;
}

case LeftArrow:
{
    if (p > -1)
    do
        p--;
    while ((lcptr->Mask[p] != '_') && (p > -1));

    break;
}

case HomeKey:
{
    p = 0;

    while (lcptr->Mask[p] != '_')
        p++;

    break;
}

```

```

case EndKey:
{
    p = lcptr->Len-1;

    while (lcptr->Mask[p] != '_')
        p--;

    break;
}

case ShiftTabKey:
case UpArrow:
{
    if (lcptr->Format != AlphaNumeric)
    {
        CleanNumberForDisplay(lcptr->WorkStr,
                               lcptr->Format);
        XYStr(lcptr->X+saylen,    lcptr->Y,
              lcptr->WorkStr);
    }

    lcptr = UpdateLcptr(lcptr);
    EndEditFlag = True;
    break;
}

case TabKey:
case DownArrow:
{
    if (lcptr->Format != AlphaNumeric)
    {
        CleanNumberForDisplay(lcptr->WorkStr,
                               lcptr->Format);
        XYStr(lcptr->X+saylen,    lcptr->Y,
              lcptr->WorkStr);
    }

    if (lcptr->Validation ?
        (*(lcptr->Validation))(lcptr->WorkStr) : 1)
    {
        lcptr = UpdateLcptr(lcptr);
        EndEditFlag = True;
    }

    break;
}

case EscKey:
{
    ExitGetFlag = True;
    break;
}

```

```

case CarriageReturn:
{
    if (lcptr->Format != AlphaNumeric)
    {
        CleanNumberForDisplay(lcptr->WorkStr,
                               lcptr->Format);
        XYStr(lcptr->X+saylen,      lcptr->Y,
              lcptr->WorkStr);
    }

    if (lcptr->Validation ?
        (*(lcptr->Validation))(lcptr->WorkStr) : 1)
    {
        EndEditFlag = True;

        if (lcptr->Execute)
            (*(lcptr->Execute))();

        if ((lcptr = UpdateLcptr(lcptr)) == NULL)
            ExitGetFlag = True;
    }
    break;
}

if (lcptr != NULL)
{
    if (    (p == lcptr->Len - 1) &&
            ((lcptr->Format == Percent) ||
             (lcptr->Format == Percent_Decimal)))
        p--;
    else
        if (p == lcptr->Len)
            p = lcptr->Len - 1;

    if (p == -1)
        if (lcptr->Format == Dollar || lcptr->Format ==
            Dollar_Cents)
            p = 1;
        else
            p = 0;
}

}
while    ((!EndEditFlag)    &&    (!ExitGetFlag)    &&
          (!CheckKeyList(g.EndKeys)));
}
while ((!ExitGetFlag) && (!CheckKeyList(g.EndKeys)));

```

```
    if (LastKey != EscKey)
    {
        lcptr = g.First;

        while (lcptr != NULL)
        {
            CleanEntry(lcptr);
            lcptr = lcptr->Next;
        }
    }

    RestoreCursorStatus();
}
```

```

/*-----
                                FreeGets

PURPOSE   :   frees memory taken up by the linked list pointed by g
              (struct GetHead) -- each node is of type GetField
ASSUMES   :   list has been created earlier
CALL      :   FreeGets(g)
INPUT PARAMETERS : (struct GetHead g)
              *g -- pointer to struct GetHead
FUNCTION CALLS( other than standard functions ) : none
RETURNS   :   nothing
SIDE EFFECTS : none
-----*/

```

```

void FreeGets(struct GetHead g)
{
    struct GetField *lcptr;

    lcptr = g.First;          /* start at the head of the list */

    /* while list end not reached, free each node */

    while (lcptr->Next != NULL)
    {
        g.First = g.First->Next;
        free(lcptr->WorkStr);
        free(lcptr->Mask);
        free(lcptr->SayStr);
        free(lcptr->DummyStr);
        free(lcptr);
        lcptr = g.First;
    }

    return;
}

```

```

/*-----
                                ErrorMessage

PURPOSE   :   prints an error message in a window
ASSUMES   :   nothing
CALL      :   ErrorMessage(*str)
INPUT PARAMETERS : (char *str)
                *str -- pointer to string to be displayed
FUNCTION CALLS( other than standard functions ) :
                ExplodeWindow(x1,.....)
                CursorOff()
                CenterText(row ,str)
                PopWindow()
RETURNS   :   nothing
SIDE EFFECTS : none
-----*/

void ErrorMessage(char *str)
{
    /* draw a slowly expanding window, with error message heading
    */
    ExplodeWindow( 2,9,77,14,
                  Single_Single,
                  RED, YELLOW, RED, YELLOW,
                  " Error ", "", True);

    CursorOff();           /* turn cursor off */
    CenterText(2, str);     /* display actual message */
    CenterText(3, "Press any key to continue.");
    getch();
    PopWindow();           /* pop the window off the screen */
}

/**   End of File   **/

```

5.0 Intersim.c

Program cover sheet

Program name : INTERSIM.C Latest Version: 1.0 Date: 08/30/91

Languages: C Manufacturer: Microsoft Version : 6.0

Description:

The program is the user interface part for the intersimulator delay study software. The header file *Delay.h* is included with it. It is executed by typing `intersim <integer>` at the DOS prompt. If `<integer>` is 0 or 99, it indicates that the packets to both ASATs will be delayed. If the `<integer>` is 1, then packets to ASAT1 only will be delayed, else if `<integer>` is 2, then packets to ASAT2 only will be delayed. If the `<integer>` is missing, then it is assumed to be 0. The program gives the user a starting trial number and then reads the desired delay values and trial duration. The ASATs are started up and the program spawns a child process to verify this. Another child process is then spawned, which takes care of collecting and delaying the network packets. Control returns to the program after all the trial results have been saved and the trial is complete (i.e., either / both ASATs have crashed or trial time is completed or a user generated interrupt has been received). The program is error proof for the following user errors:

- a) user responses for a y/n are checked.
- b) delay values are predetermined to be 0, 250, 500, 750 milliseconds. Any other delay value will not be accepted.
- c) trial duration is limited to 30 minutes (except 0). Out of range values will not be accepted.

The program will not crash under such user errors, and under normal running condition of the ASATs. In the event that the ASATs cannot be started up, the program allows the user to exit gracefully.

```

/*****
* Module      : intersim.c
* Programmer   : Sandhya Chandarlapaty
* Purpose      : This module provides a graphic user interface
*                for the Intersimulator delay study conducted
*                in the Aviation Trainer Research Laboratory on
*                the ASATs.
* Compilation  : This module is compiled using the Microsoft
*                6.0 compiler and linker.
* Usage        : intersim <option>
*                <option> can be 0,1,2,99
*                0 or 99 - delay from ASAT 1 to 2 and ASAT 2
*                        to 1
*                1      - delay from ASAT 1 to 2 only
*                2      - delay from ASAT 2 to 1 only
*                if no option is given, default is 0
* Other Info    : The module spawns the "posdelay" module. It
*                will run without the "posdelay" module, but it
*                will end with an error screen; it will still
*                exit gracefully
*
*****/

/*****
* includes
*****/

#include <stdio.h>
#include <ctype.h>
#include <graph.h>
#include <stdlib.h>
#include <string.h>
#include <process.h>
#include "delay.h"

/*****
* globals
*****/

int *asat1_2, /* pointer to delay from asat 1 to asat 2 */
    *asat2_1, /* pointer to delay from asat 2 to asat 1 */
    *trial_time /* pointer to trial time for delay study */
    trial_number = 1, /* number of the current trial */
    asat_number; /* number of the ASAT, either 1 or 2 */

```



```

/*****
* functions *
*****/

/* Forward declarations */

void hit_any(int color,char *buffer,int text);
void heading(int color);
void getstr(char buf[],int len,int x,int y);
void draw_border(int x1,int y1,int x2,int y2);
void show(int x,int y,char *buffer);
void paint(short color,short x1,short y1,short x2,short y2);

/*****
*
* trial_window()
*
*
* PURPOSE : draws the trial_window, shows trail number, gets
* 'y' or 'n' for continuation or exit
*
* ASSUMES : nothing
* CALL : trail_window(reply);
*
* INPUT PARAMETERS : reply -- pointer to character
*
* FUNCTION CALLS( other than standard functions ) :
*
* paint(color,x1,y1,x2,y2)
* heading(color);
* draw_border(x1,y1,x2,y2)
* getstr(buf,x,y)
* checker(answer,x,y)
* hit_any(color,buf,color)
* show(x,y,buf);
*
* RETURNS : nothing
*
* SIDE EFFECTS : none
*****/

void trial_window(char *reply)
{
    char buffer[60];
    char buf[2], answer;

```

```

    _setvideomode(_VRES16COLOR);          /* draw window */
    _clearscreen(_GCLEARSCREEN);
    paint(WHITE,0,0,700,600);
    heading(BLUE);
    paint(BLUE,50,130,580,270);
    draw_border(50,130,580,270);
    _displaycursor(_GCURSORON);
    _settextcolor(BRIGHTWHITE);
    _setbkcolor(_BLUE);

    sprintf(buffer,"This is trial number : %d", trial_number);
    show(11,10,buffer);
    show(14,10," Do you want to continue ?? (y / n) : ");
    _settextposition(14,55);
    getstr(buf,2,14,55);    /* get the user's reply, validate it */
    answer = buf[0];        /* copy it into answer */

    /* while answer is invalid, get the answer again */

    while(!(checker(answer,14,55)))
    {
        getstr(buf,2,14,55);
        answer = buf[0];
    }

    if(tolower(answer) == 'n')
    {
        /* User's answer is no, try to quit, return 'n' to caller
           */

        hit_any(BLUE,"I had lots of fun !! Did you
        ??",BRIGHTWHITE);
        *reply = 'n';
    }
    else
    {
        /* User's answer is yes, try to continue, return 'y' to
           caller */

        hit_any(BLUE,"Hit any key when you are ready
        ..",BRIGHTWHITE);
        *reply = 'y';
    }
}

```

```

/*****
*
*                               inform_window()
*
*
* PURPOSE   :   draws the inform_window, shows information about
*               the valid intersimulator delay study trial
*               durations. The window is drawn in the upper
*               half of the screen leaving enough room for the
*               delay_window
*
* ASSUMES   :   nothing
*
* CALL      :   inform_window();
*
* INPUT PARAMETERS : nil
*
* FUNCTION CALLS( other than standard functions ) :
*
*               paint(color,x1,y1,x2,y2)
*               draw_border(x1,y1,x2,y2)
*               show(x,y,buf);
*
* RETURNS   :   nothing
*
* SIDE EFFECTS : none
*****/

```

```

void inform_window(void)
{
    _setvideomode(_VRES16COLOR);      /* draw window */
    _clearscreen(_GCLEARSCREEN);
    paint(WHITE,0,0,700,600);
    paint(BLUE,50,5,580,98);
    draw_border(50,5,580,98);
    _settextcolor(BRIGHTWHITE);
    _setbkcolor(_BLUE);

    show(3,15,"Valid delay times in millisecs : 0 , 250 , 500 ,
              750");
    show(5,15,"Valid trial duration : 1 - 30 minutes");
    _setbkcolor(_GRAY);
}

```

```

/*****
                                delay_window1()
*
* PURPOSE   : draws the delay_window, gets the delay from
*             ASAT1 to ASAT2, and the trail duration
*
* ASSUMES   : nothing
*
* CALL      : delay_window1(asat1_2, trial_time);
*
* INPUT PARAMETERS : int *asat1_2, *trial_time
*
* FUNCTION CALLS( other than standard functions ) :
*
*             paint(color,x1,y1,x2,y2)
*             draw_border(x1,y1,x2,y2)
*             getdelay(x,y)
*             gettrial(x,y)
*             hit_any(color,buf,color)
*             show(x,y,buf);
*
* RETURNS   : nothing
*
* SIDE EFFECTS : sets globals *asat1_2, *trial_time
*****/

```

```

void delay_window1(int *asat1_2,int *trial_time)
{
    _displaycursor(_G_CURSORON);
    paint(BLUE,50,140,580,350);
    draw_border(50,140,580,350);
    _settextcolor(BRIGHTWHITE);
    _setbkcolor(_BLUE);
    show(12,25," D E L A Y   D E T A I L S ");
    show(16,10,"Enter desired delay from ASAT 1 to ASAT 2 in msec
              :");
    show(20,10,"Enter desired trial time in mins :");
    *asat1_2 = getdelay(16,65); /* get delay value, validate it*/
    *trial_time = gettrial(20,65); /* get trial time value,
                                   validate it*/
    _displaycursor(_G_CURSOROFF);
    hit_any(BLUE,"Hit any key to continue ... ",BRIGHTWHITE);
}

```

```

/*****
*                               delay_window2()                               *
*                               *                               *
* PURPOSE : draws the delay_window, gets the delay from ASAT2 *
*           to ASAT1, and the trail duration                    *
*                               *                               *
* ASSUMES : nothing                                           *
*                               *                               *
* CALL      : delay_window2(asat2_1, trial_time);             *
*                               *                               *
* INPUT PARAMETERS : int *asat2_1, *trial_time                *
*                               *                               *
* FUNCTION CALLS( other than standard functions ) :          *
*                               *                               *
*           paint(color,x1,y1,x2,y2)                          *
*           draw_border(x1,y1,x2,y2)                          *
*           getdelay(x,y)                                       *
*           gettrial(x,y)                                       *
*           hit_any(color,buf,color)                           *
*           show(x,y,buf);                                       *
*                               *                               *
* RETURNS : nothing                                           *
*                               *                               *
* SIDE EFFECTS : sets globals *asat2_1, *trial_time          *
*****/
void delay_window2(int *asat2_1,int *trial_time)
{
    _displaycursor(_G_CURSORON);
    paint(BLUE,50,140,580,350);
    draw_border(50,140,580,350);
    _settextcolor(BRIGHTWHITE);
    _setbkcolor(_BLUE);

    show(12,25," D E L A Y   D E T A I L S ");

    show(16,10,"Enter desired delay from ASAT 2 to ASAT 1 in msec
:");
    show(20,10,"Enter desired trial time in mins :");

    *asat2_1 = getdelay(16,65);      /* get delay value, validate
                                     it*/
    *trial_time = gettrial(20,65);   /* get trial time value,
                                     validate it*/
    _displaycursor(_G_CURSOROFF);

    hit_any(BLUE,"Hit any key to continue ... ",BRIGHTWHITE);
}

```

```

/*****
*
*          delay_window3()
*
* PURPOSE   :  draws the delay_window, gets the delay from ASAT1
*               to ASAT2, delay from ASAT2 to ASAT1, and the trail
*               duration
*
* ASSUMES   :  nothing
*
* CALL      :  delay_window1(asat1_2, asat2_1, trial_time);
*
* INPUT PARAMETERS :  int *asat1_2, *asat2_1,*trial_time
*
* FUNCTION CALLS( other than standard functions ) :
*
*               paint(color,x1,y1,x2,y2)
*               draw_border(x1,y1,x2,y2)
*               getdelay(x,y)
*               gettrial(x,y)
*               hit_any(color,buf,color)
*               show(x,y,buf);
*
* RETURNS   :  nothing
*
* SIDE EFFECTS :  sets globals *asat1_2,      *trial_time
*****/

```

```

void delay_window3(int *asat1_2,int *asat2_1,int *trial_time)
{
    _displaycursor(_G_CURSORON);
    paint(BLUE,50,140,580,350);
    draw_border(50,140,580,350);

    _settextcolor(BRIGHTWHITE);
    _setbkcolor(_BLUE);
    show(11,25,"_D E L A Y   D E T A I L S ");
    show(14,10,"Enter desired delay from ASAT 1 to ASAT 2 in msec
:");
    show(17,10,"Enter desired delay from ASAT 2 to ASAT 1 in msec
:");
    show(20,10,"Enter desired trial time in mins :");

    *asat1_2 = getdelay(14,65);      /* get delay value, validate
                                     it*/
    *asat2_1 = getdelay(17,65);
    *trial_time = gettrial(20,65); /* get trial time value,
                                     validate it*/

    _displaycursor(_G_CURSOROFF);
    hit_any(BLUE,"Hit any key to continue ... ",BRIGHTWHITE);
}

```

```

/*****
*                               ready_window()                               *
*
*  PURPOSE   :   draws the ready_window, requesting user to               *
*                start the ASATs.                                         *
*
*  ASSUMES   :   nothing                                                  *
*
*  CALL      :   ready_window();                                          *
*
*  INPUT PARAMETERS : none                                              *
*
*  FUNCTION CALLS( other than standard functions ) :                    *
*
*                paint(color,x1,y1,x2,y2)                                *
*                heading(color);                                          *
*                draw_border(x1,y1,x2,y2)                                *
*                show(x,y,buf);                                           *
*
*  RETURNS   :   nothing                                                  *
*
*  SIDE EFFECTS : none                                                  *
*****/

```

```

void ready_window(void)
{
    _setvideomode(_VRES16COLOR);
    _clearscreen(_GCLEARSCREEN);
    paint(DARKGRAY,0,0,700,600);
    heading(RED);
    paint(RED,50,130,580,270);
    draw_border(50,130,580,270);
    _settextcolor(31);
    _setbkcolor(_RED);
    show(11,10," Waiting for network initialization !! ");
    show(13,10," You may start the ASATs now.. ");
    show(15,10," Press Esc key if unsuccessful, else any other
                key");
}

```

```

/*****
*                               progress_window()                               *
*
* PURPOSE   :   draws the progress_window, informs user that the
*               trail is in progress and remains in view until
*               the control returns to main driver
*
* ASSUMES   :   nothing
*
* CALL      :   progress_window();
*
* INPUT PARAMETERS : none
*
* FUNCTION CALLS( other than standard functions ) :
*
*               paint(color,x1,y1,x2,y2)
*               heading(color);
*               draw_border(x1,y1,x2,y2)
*               show(x,y,buf);
*
* RETURNS   :   nothing
*
* SIDE EFFECTS : none
*****/

```

```

void progress_window(void)
{
    _setvideomode(_VRES16COLOR);
    _clearscreen(_GCLEARSCREEN);
    paint(DARKGRAY,0,0,700,600);
    heading(RED);
    paint(RED,50,130,580,290);
    draw_border(50,130,580,290);

    _settextcolor(BRIGHTWHITE);
    _setbkcolor(_RED);
    show(11,10," Trial in progress ..... ");
    show(14,10," Please wait !!");
    show(17,10," Hit the Escape key <Esc> to abort trial");
}

```



```

/*****
*                               error_window()                               *
*
* PURPOSE   :   draws the error_window, in case spawning fails           *
*
* ASSUMES   :   nothing                                                    *
*
* CALL      :   error_window();                                           *
*
* INPUT PARAMETERS : none                                                *
*
* FUNCTION CALLS( other than standard functions ) :                     *
*
*           paint(color,x1,y1,x2,y2)                                       *
*           heading(color);                                                 *
*           draw_border(x1,y1,x2,y2)                                       *
*           show(x,y,buf);                                                 *
*           hit_any(color,buf,color)                                       *
*
* RETURNS   :   nothing                                                    *
*
* SIDE EFFECTS : none                                                    *
*****/

```

```

void error_window(void)
{
    _setvideomode(_VRES16COLOR);
    _clearscreen(_GCLEARSCREEN);

    paint(DARKGRAY,0,0,700,600);
    heading(RED);
    paint(RED,50,130,580,270);
    draw_border(50,130,580,270);

    _settextcolor(31);
    _setbkcolor(_RED);
    show(11,10," ERROR!  Delay program cannot be loaded ! ");
    show(14,10," Need to abort program ... ");

    hit_any(RED, " Hit any key to abort ...",BRIGHTWHITE);
}

```

```

/*****
*                               hit_any()                               *
*
* PURPOSE   : gets a keyhit from keyboard for continuation          *
*             purposes                                              *
* ASSUMES   : nothing                                              *
*
* CALL      : hit_any(color,buffer,text);                            *
*
* INPUT PARAMETERS :  int color -- background color                *
*                   int text  -- text color                        *
*                   char *buffer -- text to be displayed           *
*
* FUNCTION CALLS( other than standard functions ) :                *
*
*                   paint(color,x1,y1,x2,y2)                       *
*                   draw_border(x1,y1,x2,y2)                       *
*                   show(x,y,buf);                                  *
*
* RETURNS   : nothing                                              *
*
* SIDE EFFECTS : none                                              *
*****/
void hit_any(int color,char *buffer,int text)
{
    _displaycursor(_GCSOROFF);
    _settextcolor(text);

    paint(color,140,400,490,450);
    draw_border(140,400,490,450);
    show(27,25,buffer);

    getch();          /* get and toss key from the keyboard */
    _setvideomode(_DEFAULTMODE);
}

```

```

/*****
*                                     heading()                                     *
*                                     *                                           *
* PURPOSE : draws the heading on the screen                                     *
*                                     *                                           *
* ASSUMES : nothing                                                             *
*                                     *                                           *
* CALL    : heading(color);                                                    *
*                                     *                                           *
* INPUT PARAMETERS : int color -- bkcolor for the heading                     *
*                                     window                                     *
* FUNCTION CALLS( other than standard functions ) :                           *
*                                     *                                           *
*                                     paint(color,x1,y1,x2,y2)                   *
*                                     draw_border(x1,y1,x2,y2)                   *
*                                     show(x,y,buf);                             *
*                                     *                                           *
* RETURNS : nothing                                                            *
*                                     *                                           *
* SIDE EFFECTS : none                                                         *
*****/

```

```

void heading(int color)
{
    char buf[50];

    paint(color,50,10,580,65);
    draw_border(50,10,580,65);
    sprintf(buf,"AVIATION TRAINER TECHNOLOGY LABORATORY");
    _settextcolor(BRIGHTWHITE);
    _setbkcolor(color);
    show(3,21,buf);
}

```

```

/*****
*
*                               message()
*
*
* PURPOSE   :   draws the message window, with information about
*               the intersimulator delay study
*
* ASSUMES   :   nothing
*
* CALL      :   message();
*
* INPUT PARAMETERS : none
*
* FUNCTION CALLS( other than standard functions ) :
*
*               paint(color,x1,y1,x2,y2)
*               heading(color);
*               draw_border(x1,y1,x2,y2)
*               show(x,y,buf);
*               hit_any(color,buf,color)
*
* RETURNS   :   nothing
*
* SIDE EFFECTS : none
*****/

```

```

void message(void)
{
    _setvideomode(_VRES16COLOR);
    _clearscreen(_GCLEARSCREEN);

    paint(DARKGRAY,0,0,700,600);
    heading(RED);
    paint(RED,50,130,580,310);
    draw_border(50,130,580,310);

    _settextcolor(BRIGHTWHITE);
    _setbkcolor(_RED);
    show(10,25,"INTERSIMULATOR DELAY STUDY");
    show(12,10,"This PC collects data packets from the ASATs and
               puts them");
    show(14,10,"back on the network after inducing a desired time
               delay.  ");
    show(16,10,"You may choose the respective delays and the trial
               time.  ");
    show(18,10,"After the run you may save the results in a file.
               ");

    hit_any(RED,"Hit any key to proceed ...", BRIGHTWHITE);
}

```

```

/*****
*                               get_delay()                               *
*
*  PURPOSE   :  gets the delay value from the keyboard, checks      *
*                for its validity, repeatedly gets the value until *
*                it gets a valid value and then returns it to the *
*                caller. Valid delay values are : 0, 250, 500, 750 *
*                msec.                                             *
*
*  ASSUMES   :  nothing                                             *
*
*  CALL      :  get_delay(x,y);                                     *
*
*  INPUT PARAMETERS :  int x,y -- where to read input from on      *
*                        screen                                       *
*
*  FUNCTION CALLS( other than standard functions ) :              *
*
*                        show(x,y,buf);                             *
*                        delay_checker(delay)                       *
*                        get_string(str,value,x,y)                  *
*
*  RETURNS   :  value of delay                                     *
*
*  SIDE EFFECTS :  none                                           *
*****/

```

```

int getdelay(int x,int y)
{
    int gotit = 0,i;
    char str[7];
    enum valid_delays delay;

```

```

while (!gotit)      /* while user has not entered a valid delay
                    */
{
    _settextposition(x,y);
    getstr(str,7,x,y); /* get user's input, six chars max
                        */

    for (i=0; i < strlen(str); i++)
    {
        if ((isalpha(str[i])) || (ispunct(str[i])) ||
            (isspace(str[i])) || (iscntrl(str[i])))
        {
            /* if string has unwanted chrs, beep, overwrite
               with blanks */

            printf("\07");
            _settextposition(x,y);
            show(x,y,"");
            gotit = 0;
            break;
        }
        else
            gotit = 1;
    }

    if (gotit)
    {
        delay = atoi(str); /* convert the string to an integer
                           value*/

        if (delaychecker (delay)) /* check if it valid value
                                   */
            gotit = 1;
        else
        {
            /* if the input delay is not the right choice,
               beep, overwrite

               with blanks, repeat */
            printf("\07");
            _settextposition(x,y);
            show(x,y,"");
            gotit = 0;
        }
    }
}

return(delay);
}

```

```

/*****
*
*                               get_str()
*
* PURPOSE :   gets a string from the keyboard, for a specified
*             length only. The user can still use the editor
*             keys such as the BackSpace key, Del key etc. It
*             will not user to enter a string longer than the
*             specified length, returns it to caller when
*             Carriage Return key is entered
*
* ASSUMES :   nothing
*
* CALL       :   get_str(buf,value,x,y);
*
* INPUT PARAMETERS :  int x,y -- where to read input from
*                     screen
*                     int value -- maximum length of string
*                     char buf -- holds the string
*
* FUNCTION CALLS( other than standard functions ) :
*                 snow(x,y,buf);
*
* RETURNS : nothing
*
* SIDE EFFECTS : none
*****/

```

```

void getstr(char buf[],int len,int x,int y)
{
    char ch, cbuf[2];
    int i,xpos,ypos;

    _settextposition(x,y);
    xpos = x;
    ypos = y;
    i = 0;

```

```

/* Repeat while CR is not pressed or length of string < len */
do
{
    ch = getch();          /* get a char from          */

    if (ch == CARRIAGE)
        buf[i] = '\0';
    else
    {
        /* char is not Carriage Return key, check if it is
           BackSpace*/

        if (ch != BACKSPACE)
        {
            /* char is not BackSpace key, save it in buf,
               display it */

            if (i < len-1)
            {
                _settextposition(xpos,ypos);
                _sprintf(cbuf,"%c",ch);
                _outtext(cbuf);
                buf[i] = ch;
                i++;
                ypos++;
            }
            else
                printf("\07");
        }
        else
        {
            /*
               char is BackSpace key, erase previous char on
               screen remove it from buf only if buf is not
               empty else beep
            */

            if (i > 0)
            {
                i--;
                ypos--;
                _settextposition(xpos,ypos);
                _show(xpos,ypos," ");
                _settextposition(xpos,ypos);
            }
            else
                printf("\07");
        }
    }
}
while (i < len && ch != CARRIAGE);

if (i == len-1)
    buf[len-1] = '\0';
}

```



```

/*****
*                               call_initsys()                               *
*
* PURPOSE   :  spawns a child process to invoke initsys.exe and
*               sends it the asat_number. If initsys.exe is not
*               present in the directory, it will return with an
*               error
*
* ASSUMES   :  nothing
*
* CALL      :  call_initsys();
*
* INPUT PARAMETERS :  none
*
* FUNCTION CALLS( other than standard functions ) :  none
*
* RETURNS   :  nothing
*
* SIDE EFFECTS :  none
*****/

```

```

void call_initsys(void)
{
    char *args[3], buffers[3][10];
    int child_return;

    args[0] = "initsys";          /* name of executable file to be
                                   initiated */
    args[1] = itoa(asat_number, buffers[1], BASE);
    args[2] = NULL;
    child_return = spawnv(P_WAIT, "initsys.exe", args);

    /* if spawning fails, exit gracefully */
    if (child_return == -1)
        error_window();
}

```

```

/*****
*
*                               call_predelay()
*
*
* PURPOSE   :  spawns a child process to invoke predelay.exe
*               and sends it the delay values, trial duration,
*               asat number. If predelay.exe is not in the
*               directory, it will return with an error
*
* ASSUMES   :  nothing
*
* CALL      :  call_predelay();
*
* INPUT PARAMETERS :  int asat1_2, asat2_1, trial_time, asatnum
*
* FUNCTION CALLS( other than standard functions ) : none
*
* RETURNS   :  nothing
*
* SIDE EFFECTS : none
*****/

```

```

void call_predelay(int asat1_2,int asat2_1,int trial_time,int
                  asatnum)
{
    char *args[7], buffers[5][10];
    int child_return;

    args[0] = "predelay";
    args[1] = itoa(asat1_2, buffers[0], BASE);
    args[2] = itoa(asat2_1, buffers[1], BASE);
    args[3] = itoa(trial_time, buffers[2], BASE);
    args[4] = itoa(trial_number, buffers[3], BASE);
    args[5] = itoa(asatnum, buffers[4], BASE);
    args[6] = NULL;
    child_return = spawnv(P_WAIT,"predelay.exe",args);

    /* if spawning fails, exit gracefully */
    if (child_return == -1)
        error_window();
}

```

```

/*****
*                                     draw_border()
*
*   PURPOSE   :   draws a double line border, inside a window
*
*   ASSUMES   :   nothing
*
*   CALL      :   draw_border(x1,y1,x2,y2);
*
*   INPUT PARAMETERS : int x1,y1,x2,y2 -- left top, right
*                           bottom pts
*   FUNCTION CALLS( other than standard functions ) : none
*
*   RETURNS   :   nothing
*
*   SIDE EFFECTS : none
*****/

```

```

void draw_border(int x1,int y1,int x2,int y2)
{
    _setcolor(BRIGHTWHITE);
    _moveto(x1+5,y1+5);
    _lineto(x2-5,y1+5);
    _lineto(x2-5,y2-5);
    _lineto(x1+5,y2-5);
    _lineto(x1+5,y1+5);
    _moveto(x1+8,y1+8);
    _lineto(x2-8,y1+8);
    _lineto(x2-8,y2-8);
    _lineto(x1+8,y2-8);
    _lineto(x1+8,y1+8);
}

```

```

/*****
*                               delaychecker()                               *
*                                                                           *
*   PURPOSE   :   checks for validity of given delay value               *
*                                                                           *
*   ASSUMES   :   nothing                                                 *
*                                                                           *
*   CALL      :   delaychecker(value);                                    *
*                                                                           *
*   INPUT PARAMETERS : int value -- delay value                          *
*                                                                           *
*   FUNCTION CALLS( other than standard functions ) : none              *
*                                                                           *
*   RETURNS   :   true or false, based on value                          *
*                                                                           *
*   SIDE EFFECTS : none                                                  *
*****/

```

```

int delaychecker(enum valid_delays value)
{
    switch (value)
    {
        case ZERO :
        case SMALL :
        case MEDIUM :
        case LARGE :
            return(1);

        default :
            return (0);
    }
}

```

```

/*****
*                               trialchecker()                               *
*                               *                                           *
*   PURPOSE   :   checks for validity of given trail duration            *
*                value                                                    *
*                               *                                           *
*   ASSUMES   :   nothing                                                  *
*                               *                                           *
*   CALL      :   trialchecker(value);                                    *
*                               *                                           *
*   INPUT PARAMETERS : int value -- trial duration value                 *
*                               *                                           *
*   FUNCTION CALLS( other than standard functions ) : none              *
*                               *                                           *
*   RETURNS   :   true or false, based on value                          *
*                               *                                           *
*   SIDE EFFECTS : none                                                  *
*****/

int trialchecker(int value)
{
    /* Trial time is not zero minutes and less than 31 minutes */
    return(0 < value && value < MAXTRIAL);
}

```

```

/*****
*                                     checker()
*
* PURPOSE   : checks if given character is a Y/y/N/n
*
* ASSUMES   : nothing
*
* CALL      : checker(answer,x,y);
*
* INPUT PARAMETERS : char answer -- char to be tested
*                   x,y -- position to print blanks on
*                   answer
*
* FUNCTION CALLS( other than standard functions ) : none
*
* RETURNS : true or false, based on answer
*
* SIDE EFFECTS : none
*****/

```

```

int checker(char answer,int x,int y)
{
    switch(toupper(answer))
    {
        case 'Y' :
        case 'N' :
            return(1);

        default :
        {
            printf("\07");
            show(x,y," ");
            _settextposition(x,y);
            return(0);
        }
    }
}

```

```

/*****
*
*                               show()
*
*   PURPOSE   :   writes text, inside a window at x,y
*
*   ASSUMES   :   nothing
*
*   CALL      :   show(x,y,buffer);
*
*   INPUT PARAMETERS :   int x,y -- location where to write
*                           text
*                           char buffer - text to write
*
*   FUNCTION CALLS( other than standard functions ) : none
*
*   RETURNS   :   nothing
*
*   SIDE EFFECTS : none
*****/

```

```

void show(int x,int y,char *buffer)
{
    _settextposition(x,y);
    _outtext(buffer);
}

```

```

/*****
*
*                               paint()
*
*   PURPOSE   :   fills a window with given color
*
*   ASSUMES   :   nothing
*
*   CALL      :   paint(color,x1,y1,x2,y2);
*
*   INPUT PARAMETERS :   int x1,y1,x2,y2 -- left top, right
*                                           bottom
*                           int color
*
*   FUNCTION CALLS( other than standard functions ) : none
*
*   RETURNS   :   nothing
*
*   SIDE EFFECTS : none
*****/

```

```

void paint(short color,short x1,short y1,short x2,short y2)
{
    _setcolor(color);
    _rectangle(_GFillInterior,x1,y1,x2,y2);
}

```

```

/*****
*      main driver      *
*****/
void main(int argc,char *argv[])
{
    char *answer; /* pointer to 'y' or 'n' continuing execution */

    /* If no option is given, treated as if parameter was 0 */

    if (argc < 2)
        asat_number = 0;          /* No option, use default */
    else
        asat_number = atoi(argv[1]); /* Check option validity */

    if (asat_number && asat_number!=1 && asat_number != 2 &&
        asat_number != 99)
    {
        printf("Parameter error. Retry !\n");
        exit(0);
    }

    /* Allocate memory for globals */
    asat1_2      = (int *) malloc(sizeof(int));
    asat2_1      = (int *) malloc(sizeof(int));
    trial_time   = (int *) malloc(sizeof(int));
    answer       = (char *) malloc(sizeof(char));

    *asat1_2 = *asat2_1 = 0;

    message();          /* display initial message */
    trial_window(answer); /* display trail window with trial
                           number */

    /* while user does not want to quit */
    while(*answer != 'n')
    {
        inform_window(); /* draw inform_window with valid
                           values */

        /* based on asat_number, draw the appropriate window */

        if (asat_number == 1)
            delay_window1(asat1_2,trial_time); /* get delay from
                                                  ASAT 1 to 2 */
        else
            if(asat_number == 2)
                delay_window2(asat2_1,trial_time); /* get delay from
                                                      ASAT 2 to 1 */
            else
                /* get gelay form ASAT 1 to 2 */
                /* get delay from ASAT 2 to 1*/
                delay_window3(asat1_2, asat2_1, trial_time);
    }
}

```



```
ready_window();
call_initsys();      /* spawn to initiate initsys.exe*/
progress_window();

/* spawn to initiate predelay.exe*/
call_predelay(*asatl_2, *asat2_1, *trial_time, asat_number);

trial_number++;      /* increment trial number */
trial_window(answer);
    }
}

/*      End of file */
```

6.0 *initsys.c*

Program Cover Sheet

Program Name: initsys.c Latest Version: 1.1 Date: 9-15-1991

Languages: C Manufacturer: Microsoft Co. Version: 5.1

Description:

This program serves as the pre-network service before the program *delay.exe* is invoked. It transfers packets of type 0 and 1 to allow the two ASAT simulators to make decisions as to who is the master simulator.

This program is invoked, with 1 optional parameter, by *intersim.exe*. This program is terminated on receipt of a packet of type 8 or 9.

```

/*****
*   initsys: Transfer packet of type 0 and 1 before the ASATs   *
*   start to fly. Called by intersim.exe.                       *
*   *                                                             *
*   Predecessor: intersim.exe                                     *
*   Successor : None                                             *
*****/

/*****
*   includes   *
*****/

#include <dos.h>
#include <stdlib.h>
#include "inet.h"

/*****
*   declarations   *
*****/

/* Asat addresses, the two simulator used in the experiment */
char far asat1_address[6] = {0x02, 0x60, 0x8c, 0x0d, 0x25, 0xea};
char far asat2_address[6] = {0x02, 0x60, 0x8c, 0x0d, 0x20, 0xf5};

/* Local PC address */
char mypcaddress[6] = {0x02, 0x60, 0x8c, 0x4b, 0x0d, 0x9b};

/*****
*   functions   *
*****/

void main(int argc, char *argv[])
{
    int i;

    int clearFlag = TRUE; /* FALSE - rcv buf block need not be
                           released
                           TRUE - rcv buf block needs to be
                           released */
    int newaddflag = FALSE; /* FALSE - pkt addr field need not to
                             be changed
                             TRUE - pkt addr field need to be
                             changed */
    int Rcvflag = TRUE; /* FALSE - not receiving packets
                        TRUE - receiving packets */

```

```

int numpkt1,numpkt2;      /* # of packets in receiving queue */
int pktlen;               /* packet length */
int pkttype;              /* packet type */
int total_packet_length; /* total # of bytes of a packet */
int number_byte;          /* # bytes of data in a packet */
int flags;                /* specifies wait or non-wait trx */
int reqid;                /* adapter id */
int nreqid;               /* returned adapter id - reserved */
int return_code;          /* for function returns */

int AsatID;               /* simulator ID */
int AsatID0 = 0;          /* trx buffer pointer pointed by
                          Pkttrxptr */
int cntlkey = FALSE;      /* initialize control key = 0 */

if (argc < 2)              /* Decide whether we're 1 ASAT, or a
                          pair */
    network_mode = TEAMMODE;
else
    network_mode = atoi(argv[1]);

init_3com_all();
cmyAddress();

flags = ADAPTERWAIT;
reqid = ADAPTERID;

/* copy the PC address to the transmitting buffer */
for (i=0; i<6; i++)
    Pkttrxptr[i] = mypcaddress[i];

cRcvFlag(Rcvflag);        /* start to receive packets */

AsatID = ASAT2;
pkttype = PKTTYPE0;

```

```

/*
    During normal operation, will receive data as packet type
    0 or 1. So process as long as we get one of these, and
    as long as user does not say to quit.
*/
*/
while (cntlkey != ESCKEY && (pkttype == PKTTYPE0 || pkttype ==
    PKTTYPE1))
{
    if (kbhit())
        cntlkey = getch();

    numpkt1 = numpkt2 = 0;

    /*
        Check each receiving buffer,
        if there is a packet, then get the packet pointer
        and length.
    */
    if (AsatID == ASAT2)
    {
        AsatID = ASAT1;
        numpkt1 = cGetNumPkt(AsatID);

        if (numpkt1 > 0)
            pktlen = cGetOnePkt(&Pkt, AsatID, clearFlag);
    }
    else
        if (AsatID == ASAT1)
        {
            AsatID = ASAT2;
            numpkt2 = cGetNumPkt(AsatID);

            if (numpkt2 > 0)
                pktlen = cGetOnePkt(&Pkt, AsatID, clearFlag);
        }
}

```

```

/*
    if a packet arrived,
    copy data from the receiving buffer to the
    transmitting buffer and then retransmit the packet.
*/
if (pktlen > 0)
{
    pkttype = Pkt->inp[ASATTYPEBYTE];

    /* copy address from the receiving to the
       transmitting buffer */

    for (i=6; i<pktlen; i++)
        Pkttrxp[tr] = Pkt->inp[i];

    pktlen = (int) max(pktlen, MINLEN);
    total_packet_length = number_byte = pktlen;

    /* re-transmit a packet */

    return_code=cXmit1( total_packet_length,
        number_byte,
        flags,
        reqid,
        Pkttrxp[tr],
        &nreqid,
        AsatID0,
        newaddflag);
    pktlen = 0;
}
}

return_code=cResetAdapter();
}

```

7.0 Predelay.c

Program Cover Sheet

Program Name: predelay.c Latest Version: 1.1 Date: 9-15-1991

Languages: C Manufacturer: Microsoft Co. Version: 5.1

Description:

This program serves the computing of the time delay thresholds from the passing parameters from the program *intersim.exe*. The parameters received from *intersim.exe* are: delay of ASAT #1, delay of ASAT#2, trial time, trial number, and network option number. Then, this program passes the following parameters to the program *delay.exe*: delay of ASAT #1, delay of ASAT #2, trial time, trial number, an optional number, time threshold #1, and time threshold #2.

```

/*****
*
*   predelay.c :   This program receives arguments from
*                   intersim.exe. It converts the delays from
*                   milliseconds to a timer count and computes
*                   the time error thresholds. The computed
*                   values are passed to the program delay.exe.
*
*   Predecessor: calling program - intersim.exe
*   Successor   : called program delay.exe
*
*****/
```

```

/*****
*   Includes   *
*****/

/* Standard include files */

#include <graph.h>
#include <process.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/*****
/* Constants   *
*****/

#define ASAT1          1          /* ID for simulator #1   */
#define ASAT2          2          /* ID for simulator #2   */
#define MILISEC        0.838*1191.5636 /* Time counter conversion
const                               */
#define FACTOR1        1.2        /* Correction factor for
delay #1                          */
#define FACTOR2        1.2        /* Correction factor for
delay #2                          */
#define PRECISION      5          /* Precision for converting
a float                          */
#define RADIX          10        /* Base of integer      */

/*****
*   Arguments passed from intersim.exe:
*
*   argv[1] = delay on simulator #1 in milliseconds
*   argv[2] = delay on simulator #2 in milliseconds
*   argv[3] = trial time in minutes
*   argv[4] = trial number
*   argv[5] = network mode, single or two simulators
*****/

void main(int argc, char *argv[])
{
    char *args[10];      /* Converted arguments */

    char buffer[10][15]; /* conversion buf for
int->string, unsigned long->string,
float->string */

    int network_mode;    /* team mode or solitaire mode */

    /* variables for both simulator #1 and #2 */

    int    delay1,delay2; /* delay in milliseconds */
    float  delaymili1,delaymili2; /* delay in milliseconds */
    float  threshold1,threshold2; /* bottom limit of time
boundary */

    unsigned long delaytime1,delaytime2; /* delay in timer count */

```



```

if (argc < 6)
{
    printf("Too few arguments passed from %s\n",argv[0]);
    exit(0);
}

/* arguments coming from intersim.exe */

delay1 = atoi(argv[1]);      /* delay on simulator #1 in ms */
delay2 = atoi(argv[2]);      /* delay on simulator #2 in ms */
network_mode = atoi(argv[5]); /* team mode or solitaire mode */

delaytime1 = delaytime2 = 0;
delaymili1 = delaymili2 = 0;

delaymili1 = delay1*FACTOR1;
delaymili2 = delay2*FACTOR2;
delaytime1 = (unsigned long) delaymili1*MILISEC;
delaytime2 = (unsigned long) delaymili2*MILISEC;

/* initialize the delay thresholds */

threshold1 = (float) delaytime1*0.006;
threshold2 = (float) delaytime2*0.006;

/* optimize the delay thresholds */

if (delay1 == 0 && delay2 == 250)
    threshold2 = (float) delaytime2*0.005;

if (delay1 == 250 && delay2 == 0)
    threshold1 = (float) delaytime1*0.004;

if (delay1 == 250 && delay2 == 500)
{
    threshold1 = (float) delaytime1*0.005;
    threshold2 = (float) delaytime2*0.005;
}

if (delay1 == 500 && delay2 == 250)
{
    threshold1 = (float) delaytime1*0.004;
    threshold2 = (float) delaytime2*0.0035;
}

if (delay1 == 250 && delay2 == 750)
{
    threshold1 = (float) delaytime1*0.004;
    threshold2 = (float) delaytime2*0.003;
}

```

```

if (delay1 == 750 && delay2 == 250)
{
    threshold1 = (float) delaytime1*0.003;
    threshold2 = (float) delaytime2*0.004;
}

/* prepare arguments passing to delay.exe */

args[0] = "delay";

if (network_mode)
    args[1] = argv[1];
else
    args[1] = "0";

args[2] = ultoa(delaytime1,buffer[2],RADIX);
args[3] = ultoa(delaytime2,buffer[3],RADIX);
args[4] = gcvt((double)threshold1,PRECISION,buffer[4]);
args[5] = gcvt((double)threshold2,PRECISION,buffer[5]);

args[6] = argv[3];          /* trial time in minutes */
args[7] = argv[4];          /* trial number */
args[8] = argv[1];          /* delay on simulator #1 */
args[9] = argv[2];          /* delay on simulator #2 */
args[10] = NULL;

/* Call delay.exe */

if (spawnv(P_WAIT,"delay.exe",args) < 0)
    printf("Error !! cannot invoke saver.exe");
}

```

8.0 Delay.c

Program Cover Sheet

Program Name: delay.c Latest Version: 1.1 Date: 9-15-1991

Languages: C Manufacturer: Microsoft Co. Version: 5.1

Description:

This program serves as the packet controller. It receives packets from ASAT #1 and ASAT #2 simulators through the Ethernet and replaces the first six bytes of each packet with the PC's address. Then each packet is retransmitted after a preset time. The functions of this program are:

Receiving:

1. Checks byte 8 of each packet in order to switch between the receiving buffers.
2. Saves the time stamp of each received packet.

Transmitting:

1. Gets one packet from either receiving buffer and the packet's time stamp.
2. Replaces the ASAT address with the PC's network address.
3. Holds the packet for a preset time and then retransmits the packet.

Network service may be terminated by:

1. time expiration. Duration is specified by the user.
2. either ASAT simulator terminating (crashed, kill, or aborted).
3. user request.

The program receives parameters passed from *predelay.exe* and transfers some statistics to the program *posdelay.exe*. This program is called by the program *posdelay.exe* and it calls the program *posdelay.exe*.

At the end of the network services, the program transmits a type 10 packet to tell the other users that the network service is ended.

```

/*****
*
*      DELAY.C
*
*      Description:   This file contains the code which calls the
*                    functions provided by the 503.lib to
*                    receive/transmit packets through 3COM
*                    EtherLinkII board.
*
*                    This program prepares two queues to
*                    receive/transmit packets for ASAT #1 and
*                    ASAT #2.
*
*                    Packets coming from ASAT #1 and ASAT #2 are
*                    queued. Each packet is kept in the receiving
*                    buffer for the required time period. Then,
*                    the packet is retransmitted and its address
*                    is changed to the required target address.
*
*      Predecessor:   calling program predelay.exe
*      Successor  :   called program posdelay.exe
*****/

/*****
*      Includes      *
*****/

/* standard include files */

#include <dos.h>
#include <process.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

/* Intersimulator include file */

#include "inet.h"

/*****
*      Forward Declarations  *
*****/

void init_parms(void);
unsigned long get_diff_time();

/*****
*      Constants      *
*****/

#define PRECISION      0      /* precision of converting a float */
#define RADIX          10     /* base of integer */

```

```

/*****
/* Declarations *
*****/

/* ASAT addresses, the two simulator used in the experiment */

char far asat1_address[6] = {0x02, 0x60, 0x8c, 0x0d, 0x25, 0xea};
char far asat2_address[6] = {0x02, 0x60, 0x8c, 0x0d, 0x20, 0xf5};

/* enforced PC address, which can be any address */

char mypcaddress[6] = {0x02, 0x60, 0x8c, 0x4b, 0x0d, 0x9b};

/* Global variables */

int clearFlag = 0;      /* 0 - rcv buffer block need not to be
                        1 - rcv buffer block need to be
                           released */

int newaddflag = 0;     /* 0 - packet address field need not to be
                        1 - packet address field need to be
                           changed */

int Rcvflag = 0;        /* 0 - not receive packets
                        1 - receive packets */

int minutes,seconds;    /* trial time tracing variables */

/* variables for both simulator #1 and #2 */

int errorcount1,errorcount2; /* count for time stamping errors */
int first1,first2;          /* first packet receiving flag */
int maxinQ1,maxinQ2;        /* max. # of packet in rcv queue */
int mininQ1,mininQ2;        /* min. # of packet in rcv queue */
int minus1,minus2;          /* # of packets trx below time line */
int outflag1,outflag2;      /* negative time error summation flg*/
int plus1,plus2;            /* # of packets trx above time line */
int timeout1,timeout2;      /* packet holding time out flag */
int trxstatus1,trxstatus2;   /* packet holding flag */

double          summinus1,summinus2; /* sum of negative time
                                     error */
double          sumplus1,sumplus2;   /* sum of positive time
                                     error */
double          sumt1,sumt2;         /* sum of total time error */
unsigned long    icnt1,icnt2;        /* packet receiving counter */
unsigned long    trxcnt1,trxcnt2;    /* packet transmitting
                                     counter */

```

```

/*****
*   Arguments passed from predelay.exe:
*
*   argv[1] = single or two simulators code
*   argv[2] = delay on simulator #1 in timer tick count
*   argv[3] = delay on simulator #2 in timer tick count
*   argv[4] = time error threshold for simulator #1
*   argv[5] = time error threshold for simulator #2
*   argv[6] = trial time in minutes
*   argv[7] = trial number
*   argv[8] = delay on simulator #1 in milliseconds
*   argv[9] = delay on simulator #2 in milliseconds
*****/

void main(int argc, char *argv[])
{
    char *args[17];          /* arguments passing pointer array */
    char *stopstr;           /* data converting stop string pointer */

    /*
     data converting buffers, int->string, unsigned
     long->string, float->string
    */

    char buffer[17][10];

    int i;
    int trxflag;              /* trx packet id, 1=Asat1, 2=Asat2 */
    int numpkt, pktlen;        /* # packet, packet length */
    int total_packet_length;  /* # bytes of a packet */
    int number_byte;          /* # bytes of a packet */
    int flags;                /* specifies wait or non-wait trx */
    int reqid;                /* adapter id */
    int nreqid;               /* returned adapter id, reserved */
    int return_code;          /* return value of a function call */
    int AsatID;               /* simulator ID */
    int AsatID0 = 0;          /* trx buf ptr pointed by Pktrrxptr */
    int endrun = FALSE;       /* network service control flag */
    int cntlkey = FALSE;      /* assign control key not = ESCKEY */
    float timef;              /* time difference in count */
    int minute, second;
    int netflag;              /* network termination status */

    /* 0 - normal time end */
    /* 1 - simulator #1 is dead */
    /* 2 - simulator #2 is dead */
    /* 9 - ended by user */

```

```

/* variables for both simulator #1 and #2 */

int numpkt1,numpkt2;    /* # of packet in receiving queue */
int pktlen1,pktlen2;    /* packet length */
int round1,round2;      /* time counter rounding flag */

float threshold1,threshold2; /* time allowance below time line*/

unsigned long asaterror1,asaterror2; /* time stamping error
                                     compensation */
unsigned long Atime1,Atime2,Atime3; /* time stamping for Asat
                                     #1 */
unsigned long Btime1,Btime2,Btime3; /* time stamping for Asat
                                     #2 */
unsigned long delaytime1,delaytime2; /* packet holding time in
                                     count */
unsigned long nexttime1,nexttime2; /* next packet transmitting
                                     time */
unsigned long timecount,timedif;

/* get the arguments passed by predelay.exe */

if (argc < 10)
{
    printf("Less arguments passed from %s\n",argv[0]);
    exit(0);
}

network_mode = atoi(argv[1]);

delaytime1 = strtoul(argv[2],&stopstr,0);
delaytime2 = strtoul(argv[3],&stopstr,0);

threshold1 = (float) strtod(argv[4],&stopstr)/100000;
threshold2 = (float) strtod(argv[5],&stopstr)/100000;

minutes = atoi(argv[6]);

seconds = 0;
minute = 0;

init_3com_all();
cmyAddress();

/* copy the PC address to the transmitting buffer */

for (i=0; i<6; i++)
    Pktrxp[tr[i]] = mypcaddress[i];

flags = ADAPTERWAIT;
reqid = ADAPTERID;

/* compute the time stamping error compensation */

asaterror1 = (unsigned long) ASATERROR1*MILISEC;
asaterror2 = (unsigned long) ASATERROR2*MILISEC;

```

```

init_parms();                /* initialize global variables */
cRcvFlag(Rcvflag);           /* start to receive packets */
_dos_gettime(&timein);       /* get network starting time */
second = timein.second;

clearFlag = 1;               /* clear receiving buffer after read */
AsatID = ASAT2;

/*
    Loop, terminated by:
        ESC key,
        time end,
        simulator #1 termination or
        simulator #2 termination
*/
while ((cntlkey != ESCKEY || numpkt1 > 0 || numpkt2 > 0) &&
        !endrun)
{
    if (kbhit()) cntlkey = getch();
    if (cntlkey == ESCKEY) netflag = USERSTOP;
    if (cntlkey == ESCKEY || endrun)
    {
        Rcvflag = FALSE;
        cRcvFlag(Rcvflag);      /* stop receive packets */
    }

    endrun = check_time(&second); /* check end of time */

    if (endrun)
        netflag = NORMALEND;
}

```



```

/*
    get the packet counter in the receiving buffer from
    either simulator
*/

numpkt1 = numpkt2 = 0;

if (AsatID == ASAT2)
{
    AsatID = ASAT1;
    numpkt = numpkt1 = cGetNumPkt(AsatID);

    if (numpkt > 1)
        first1 = FALSE;
    else
        first1 = TRUE;

    if (numpkt > maxinQ1)
        maxinQ1 = numpkt;

    if (numpkt < mininQ1 && numpkt > 0 && icnt1 > 10 &&
        !kbhit())
        mininQ1 = numpkt;
}
else
if (AsatID == ASAT1)
{
    AsatID = ASAT2;
    numpkt = numpkt2 = cGetNumPkt(AsatID);

    if (numpkt > 1)
        first2 = FALSE;
    else
        first2 = TRUE;

    if (numpkt > maxinQ2)
        maxinQ2 = numpkt;

    if (numpkt < mininQ2 && numpkt > 0 && icnt2 > 10 &&
        !kbhit())
        mininQ2 = numpkt;
}

```

```

/* in case of the receiving buffers are not empty */
if (numpkt1 > 0 || numpkt2 > 0)
{
    if ((!trxstatus1 && AsatID == ASAT1) ||
        (!trxstatus2 && AsatID == ASAT2))
    {
        /*
            get time stamp, check time stamp errors, and
            compute the packet transmitting times
        */
        if (clearFlag)
        {
            return_code =
                cGetTimeQPtr(AsatID,&timeQ1,&timeQ2);

            if (AsatID == ASAT1 && !trxstatus1)
            {
                if (first1)
                {
                    first1 = FALSE;
                    Atime1 = Atime2 = Atime3 = *timeQ1;
                }
                else
                {
                    Atime3 = Atime2;
                    Atime2 = Atime1;
                    Atime1 = *timeQ1;
                }

                if (numpkt1 > 1 && Atime1 < Atime2 &&
                    Atime1 > Atime3)
                {
                    errorcount1++;
                    nexttime1 =
                        Atime2+delaytime1+asaterror1;
                }
                else
                    nexttime1 = *timeQ1+delaytime1;

                if (nexttime1 < *timeQ1)
                    round1 = TRUE;
                else
                    round1 = FALSE;
            }
        }
    }
}

```

```

if (AsatID == ASAT2 && !trxstatus2)
{
    if (first2)
    {
        first2 = FALSE;
        Btime1 = Btime2 = Btime3 = *timeQ1;
    }
    else
    {
        Btime3 = Btime2;
        Btime2 = Btime1;
        Btime1 = *timeQ1;
    }

    if (numpkt2 > 1 && Btime1 < Btime2 &&
        Btime1 > Btime3)
    {
        errorcount2++;
        nexttime2 =
            Btime2+delaytime2+asaterror2;
    }
    else
    {
        nexttime2 = *timeQ1+delaytime2;
    }

    if (nexttime2 < *timeQ1)
        round2 = TRUE;
    else
        round2 = FALSE;
}
}

/* get 1 packet from receiving buffer of either
   simulator */

pktlen = 0;

if ((AsatID == ASAT1 && !trxstatus1 && numpkt1) ||
    (AsatID == ASAT2 && !trxstatus2 && numpkt2))
{
    pktlen = cGetOnePkt(&Pkt,AsatID,clearFlag);

    /* if packet length exceeds the max length
       setting */

    if (pktlen > MAXLEN)
    {
        pktlen = MAXLEN;
        printf("Maximum packet length setting
            error\n");
    }
}

```

```

/*
    setup
        trx flag,
        packet lenth,
        receiving count,
        packet pointer, and
        check the simulator state for
        simulator #1
*/
if (AsatID == ASAT1 && pktlen > 0)
{
    trxstatus1 = TRUE;
    pktlen1 = pktlen;
    icnt1++;
    Pkt1 = Pkt;

    /* check if simulator #1 is dead */

    /* pktlen > 50 guarantees the packet type
       is not type 0 or 1 */

    if (Pkt->inp[ASATSTATEBYTE] & DEADMARK &&
        pktlen > 50)
    {
        endrun = TRUE;
        netflag = ASAT1DEAD;
    }
}

/*
    setup
        trx flag,
        packet lenth,
        receiving count,
        packet pointer, and
        check the simulator state for the
        simulator #2
*/

```

```

        if (AsatID == ASAT2 && pktlen > 0)
        {
            trxstatus2 = TRUE;
            pktlen2 = pktlen;
            icnt2++;
            Pkt2 = Pkt;

            /*
             check if simulator #2 is dead
             pktlen > 50 guarantees the packet type
             is not type 0 or 1
            */

            if (Pkt->inp[ASATSTATEBYTE] & DEADMARK &&
                pktlen > 50)
            {
                endrun = TRUE;
                netflag = ASAT2DEAD;
            }
        }
    }

    /* in case there is a packet to be transmitted */
    if (trxstatus1 || trxstatus2)
    {
        cGetTimeCount();
        timecount = *timeptr;

        /* check time duration for packet coming from
           simulator #1 */

        if (trxstatus1 && !timeout1)
        {
            if (delaytime1 == 0)
                timeout1 = TRUE;

            if (timecount < Atime1 && timecount > nexttime1
                && round1)
                timeout1 = TRUE;
            else
                if (timecount > nexttime1 && !round1)
                    timeout1 = TRUE;
        }
    }

```

```

if (timeout1 && !outflag1)
{
    outflag1 = TRUE;
                                t i m e d i f    =
        get_diff_time(nexttime1,timecount);
    plus1++;

    if (timecount > nexttime1)
    {
        sumt1 += (double) timedif;
        sumplus1 += (double) timedif;
    }
}
else
if (!timeout1 && !outflag1)
{
                                t i m e d i f    =
        get_diff_time(timecount,nexttime1);
    timef = (float) timedif;

    if (timef < threshold1)
    {
        outflag1 = TRUE;
        timeout1 = TRUE;
        minus1++;
        summinus1 += (double) timedif;
        sumt1 -= (double) timedif;
    }
}
}

/* check the time duration for packet from
simulator #2 */

if (trxstatus2 && !timeout2)
{
    if (delaytime2 == 0)
        timeout2 = TRUE;

    if (timecount < Btime1 && timecount > nexttime2
        && round2)
        timeout2 = TRUE;
    else
    if (timecount > nexttime2 && !round2)
        timeout2 = TRUE;
}

```

```

if (timeout2 && !outflag2)
{
    outflag2 = TRUE;
                                t i m e d i f =
        get_diff_time(nexttime2,timecount);
    plus2++;

    if (timecount > nexttime2)
    {
        sumt2 += (double) timedif;
        sumplus2 += (double) timedif;
    }
}
else
if (!timeout2 && !outflag2)
{
                                t i m e d i f =
        get_diff_time(timecount,nexttime2);
    timef = (float) timedif;

    if (timef < threshold2)
    {
        outflag2 = TRUE;
        timeout2 = TRUE;
        minus2++;
        summinus2 += (double) timedif;
        sumt2 -= (double) timedif;
    }
}
}

```

```

/* Preparing and transmitting a packet */
if (timeout1 || timeout2)
{
    if (timeout1 && trxstatus1)
    {
        /* copy from receiving buffer to
           transmitting buffer */

        for (i=6; i<pktlen1; i++)
            Pkttrxptr[i] = Pkt1->inp[i];

        pktlen1 = (int) max(pktlen1, MINLEN);
        total_packet_length = number_byte =
                                pktlen1;
        trxflag = ASAT1;
    }
    else
    if (timeout2 && trxstatus2)
    {
        /* copy from the receiving to transmitting
           buffer */

        for (i=6; i<pktlen2; i++)
            Pkttrxptr[i] = Pkt2->inp[i];

        pktlen2 = (int) max(pktlen2, MINLEN);
        total_packet_length = number_byte =
                                pktlen2;
        trxflag = ASAT2;
    }
    if (timeout1 || timeout2)
    {
        return_code = cXmit1( total_packet_length,
                                number_byte,
                                flags,
                                reqid,
                                Pkttrxptr,
                                &nreqid,
                                AsatID0,
                                newaddflag);
    }
}

```



```

        if (trxflag == ASAT1)
        {
            trxcnt1++;
            trxstatus1 = FALSE;
            timeout1    = FALSE;
            outflag1    = FALSE;
        }
        else
        if (trxflag == ASAT2)
        {
            trxcnt2++;
            trxstatus2 = FALSE;
            timeout2    = FALSE;
            outflag2    = FALSE;
        }
    }
}

/*
    Now, trial is ended. It needs to broadcast the ending
    information by sending a packet type 10 using each Asat
    simulator's address
*/

Pkttrxptr[ASATTYPEBYTE] = PKTTYPE10;
total_packet_length = number_byte = 294;

for (i=0; i<6; i++)          /* copy Asat #1 address */
    Pkttrxptr[i] = asat1_address[i];

return_code = cXmit1(    total_packet_length,
                        number_byte,
                        flags,
                        reqid,
                        Pkttrxptr,
                        &nreqid,
                        AsatID0,
                        newaddflag);

for (i=0; i<6; i++)          /* copy Asat #2 address */
    Pkttrxptr[i] = asat2_address[i];

```

```

return_code = cXmit1(    total_packet_length,
                        number_byte,
                        flags,
                        regid,
                        Pkttrxptra,
                        &nregid,
                        AsatID0,
                        newaddflag);

return_code=cResetAdapter(); /* reset Ether Link II adapter */

/* prepare for data passing to the program 'posdelay.c' */

args[0]      = "posdelay";
args[1]      = ultoa(trxcnt1,buffer[1],RADIX);
args[2]      = ultoa(trxcnt2,buffer[2],RADIX);

args[3]      = itoa(plus1,buffer[3],RADIX);
args[4]      = itoa(plus2,buffer[4],RADIX);
args[5]      = itoa(minus1,buffer[5],RADIX);
args[6]      = itoa(minus2,buffer[6],RADIX);

args[7]      = gcvt(sumplus1,PRECISION,buffer[7]);
args[8]      = gcvt(sumplus2,PRECISION,buffer[8]);
args[9]      = gcvt(summinus1,PRECISION,buffer[9]);
args[10]     = gcvt(summinus2,PRECISION,buffer[10]);
args[11]     = itoa(seconds,buffer[11],RADIX);

args[12]     = argv[7];                /* trial number */
args[13]     = argv[8];                /* delay on Asat #1 */
args[14]     = argv[9];                /* delay on Asat #2 */
args[15]     = itoa(netflag,buffer[15],RADIX);
args[16]     = NULL;

/* Call posdelay.exe */

if (return_code=spawnv(P_WAIT,"posdelay.exe",args) < 0)
{
    printf("Spawn error: %d for program\n",return_code,args[0]);
}
}

```

```

/*****
* init_parms: Initialize the global variables.
* Return      : None
*****/

```

```

void init_parms(void)
{
    newaddflag = FALSE;
    first1     = first2     = TRUE;
    outflag1   = outflag2   = FALSE;
    timeout1   = timeout2   = FALSE;
    trxstatus1 = trxstatus2 = FALSE;
    Rcvflag    = TRUE;

    errorcount1 = errorcount2 = 0;
    icnt1       = icnt2       = 0;
    maxinQ1     = maxinQ2     = 0;
    mininQ1     = mininQ2     = 99;
    minus1      = minus2      = 0;
    plus1       = plus2       = 0;
    sumplus1    = sumplus2    = 0;
    summinus1   = summinus2   = 0;
    sumt1       = sumt2       = 0;
    trxcnt1     = trxcnt2     = 0;
}

```

```

/*****
* get_diff_time: get the difference between two time counter
*               readings.
* Return        : elapsed time reading
*****/

```

```

unsigned long get_diff_time(unsigned long time1,unsigned long
time2)
{
    unsigned long devtime,differ,remainder;
    double t1,t2;

    if (time2 > time1)
        differ = time2-time1;
    else
    {
        t1 = (double) time1/256/256;
        t2 = t1*256*256;
        remainder = time1-(unsigned long) t2;
        t1 = (double) (256*256-1)-t1+1;
        t2 = t1*256*256+remainder;
        devtime = (unsigned long) t2;
        differ = devtime+time2;
    }

    return(differ);
}

```

```

/*****
* check_time: check the trial time to be ended.
* Return      : 0 - trial time is not ended.
*              1 - trial time is ended.
*****/

```

```

int check_time(int *second)
{
    int min;

    _dos_gettime(&timein);

    if (timein.second != *second)
    {
        seconds++;
        *second = timein.second;
    }

    min = (int) seconds/60;

    if (min < minutes)
        return(0);
    else
        return(1);
}

```

9.0 Posdelay.c

Program Cover Sheet

Program Name: posdelay.c Latest Version: 1.1 Date: 9-15-1991

Languages: C Manufacturer: Microsoft Co. Version: 5.1

Description:

This program serves to pass and compute parameters associated with data saving. It receives parameters from the program *delay.exe* and passes the parameters to the program *saver.exe*. The computations performed in this program include the average time delay error and the time delay error range for each ASAT simulator.

```

/*****
*
*   posdelay.c : A program receives parameters from delay.exe,
*               computes the average delay errors, then passes
*               parameters to saver.exe.
*   Predecessor: calling program - delay.exe
*   Successor  : called program - saver.exe
*
*****/
```

```

/*****
*   Includes   *
*****/

/* standard include files */

#include <graph.h>
#include <process.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

/*****
/* Constants   *
*****/

#define RADIX      10      /* base of integer */
#define PRECISION  10      /* precision of converting a float # */
#define MILLISEC   0.838*1191.5636 /* time counter reading
                                   conversion */

/*****
*   Arguments passed from delay.exe:
*
*   argv[1] = total # packets retransmitted for simulator #1
*   argv[2] = total # packets retransmitted for simulator #2
*   argv[3] = total # packets retransmitted above the time
*           boundary for S. #1
*   argv[4] = total # packets retransmitted above the time
*           boundary for S. #2
*   argv[5] = total # packets retransmitted below the time
*           boundary for S. #1
*   argv[6] = total # packets retransmitted below the time
*           boundary for S. #2
*   argv[7] = total delay error above the time boundary for
*           simulator #1
*   argv[8] = total delay error above the time boundary for
*           simulator #2
*   argv[9] = total delay error below the time boundary for
*           simulator #1
*   argv[10] = total delay error below the time boundary for
*           simulator #2
*   argv[11] = actual trial time in seconds
*   argv[12] = trial number
*   argv[13] = delay time on simulator #1 in milliseconds
*   argv[14] = delay time on simulator #2 in milliseconds
*   argv[15] = network termination flag
*****/

```

```

void main(int argc, char *argv[])
{
    int i;

    /*
        data converting buffers,
        int->string,
        unsigned long->string,
        float->string
    */

    char *args[19];          /* arguments passing pointer array */
    char buffer[19][15];     /* data converting buffers */

    char *stopstr;           /* data converting stop string pointer*/

    /* variables for both simulator #1 and #2 */

    int minus1;              /* # packets sent to Asat #1 below time
                               boundary */
    int minus2;              /* # packets sent to Asat #2 below time
                               boundary */
    int plus1;               /* # packets sent to Asat #1 above time
                               boundary */
    int plus2;               /* # packets sent to Asat #2 above time
                               boundary */

    double aveminus1;        /* ave error below time boundary for Asat
                               #1 */
    double aveminus2;        /* ave error below time boundary for Asat
                               #2 */
    double aveplus1;         /* ave error above time boundary for Asat
                               #1 */
    double aveplus2;         /* ave error above time boundary for Asat
                               #2 */
    double avet1;            /* total average delay error for Asat #1 */
    double avet2;            /* total average delay error for Asat #1 */
    double sumt1;            /* total delay error for Asat #1 */
    double sumt2;            /* total delay error for Asat #2 */
    double summinus1;        /* total delay error below time boundary
                               for #1 */
    double summinus2;        /* total delay error below time boundary
                               for #2 */
    double sumplus1;         /* total delay error above time boundary
                               for #1 */
    double sumplus2;         /* total delay error above time boundary
                               for #2 */
    unsigned long trxcnt1;    /* total packets transmitted to
                               Asat #1 */
    unsigned long trxcnt2;    /* total packets transmitted to
                               Asat #2 */

    /* get the arguments passed by 'delay.exe' */

    if (argc < 16)
    {

```

```
        printf("Too few arguments passed to %s\n",argv[0]);
        exit(0);
    }

    trxcnt1 = strtoul(argv[1],&stopstr,0);
    trxcnt2 = strtoul(argv[2],&stopstr,0);

    plus1 = atoi(argv[3]);
    plus2 = atoi(argv[4]);
```



```

minus1 = atoi(argv[5]);
minus2 = atoi(argv[6]);

sumplus1 = strtod(argv[7], &stopstr);
sumplus2 = strtod(argv[8], &stopstr);

summinus1 = strtod(argv[9], &stopstr);
summinus2 = strtod(argv[10], &stopstr);

sumt1 = sumplus1 - summinus1;
sumt2 = sumplus2 - summinus2;

if (mininQ1 == 99)
    mininQ1 = 0;

if (mininQ2 == 99)
    mininQ2 = 0;

if (trxcnt1 > 0)
    avet1 = sumt1 / (double) (trxcnt1 * MILLISEC);
else
    avet1 = 0;

if (plus1 > 0)
    aveplus1 = (double) sumplus1 / (double) (plus1 * MILLISEC);
else
    aveplus1 = 0;

if (minus1 > 0)
    aveminus1 = (double) summinus1 / (double) (minus1 * MILLISEC);
else
    aveminus1 = 0;

if (trxcnt2 > 0)
    avet2 = sumt2 / (double) (trxcnt2 * MILLISEC);
else
    avet2 = 0;

if (plus2 > 0)
    aveplus2 = (double) sumplus2 / (double) (plus2 * MILLISEC);
else
    aveplus2 = 0;

if (minus2 > 0)
    aveminus2 = (double) summinus2 / (double) (minus2 * MILLISEC);
else
    aveminus2 = 0;

```

```

/* Prepare arguments to pass to 'saver.exe' */

args[0] = "saver";
args[1] = argv[13];          /* delay on simulator #1 in
                             milliseconds */
args[2] = argv[14];          /* delay on simulator #2 in
                             milliseconds */
args[3] = argv[11];          /* actual trial time in seconds */
args[4] = ultoa(trxcnt1,buffer[4],RADIX);
args[5] = ultoa(trxcnt2,buffer[5],RADIX);
args[6] = itoa(plus1,buffer[6],RADIX);
args[7] = itoa(plus2,buffer[7],RADIX);
args[8] = itoa(minus1,buffer[8],RADIX);
args[9] = itoa(minus2,buffer[9],RADIX);
args[10] = gcvrt(aveplus1,PRECISION,buffer[10]);
args[11] = gcvrt(avep'us2,PRECISION,buffer[11]);
args[12] = gcvrt(aveminus1,PRECISION,buffer[12]);
args[13] = gcvrt(aveminus2,PRECISION,buffer[13]);
args[14] = gcvrt(avet1,PRECISION,buffer[14]);
args[15] = gcvrt(avet2,PRECISION,buffer[15]);
args[16] = argv[12];          /* trial number */
args[17] = argv[15];          /* network termination flag */
args[18] = NULL;

for (i=0; i<3; i++)
    printf("\07");

/* Call saver.exe */

if (spawnv(P_WAIT, "saver.exe",args) < 0)
    printf("Error !! cannot invoke saver.exe");
}

```

Program cover sheet

Program name : SAVER.C Latest Version: 1.0 Date: 08/30/91Languages: C Manufacturer: Microsoft Version : 6.0

Description:

The program is the user interface of the intersimulator delay study software. The header file *Delay.h* is included with it. It is not directly executed from DOS. Instead it is spawned as a process by *predelay.exe* and receives all the test parameters and test results of a trial from *predelay.exe*. The program informs the user when the trial has been completed and displays the results on the screen. The program then asks the user if the data is to be saved. If so, the user must supply a file name. If the specified file name is already in use, *saver.exe* asks for an alternate file name or permission to overwrite the existing file. All file names are limited to 8 characters and will be created with a .LOG extension in the current directory. After the data is saved, control returns to *predelay.exe*, which in turn returns to *intersim.exe*. The program validates user responses.

```

/*****
*   Module       : saver.c
*
*   Programmer   : Sandhya Chandarlapaty
*
*   Purpose      : This module provides a graphic user interface
*                  for the Intersimulator delay study conducted
*                  in the Aviation Trainer Research Laboratory.
*
*   Compilation  : This module must be compiled using the
*                  Microsoft 6.0 compiler and linker.
*
*   Usage        : spawned by posdelay.c
*
*   Other Info   : The module is spawned by the posdelay module.
*                  It saves the trial results to a file and
*                  displays them on the screen. It returns to
*                  intersim.c
*****/

```

```

/*****
* includes *
*****/

#include <stdio.h>
#include <ctype.h>
#include <graph.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <process.h>
#include "delay.h"

/*****
* globals *
*****/

int          trial_number, /* current trail number */
             asat1_2,      /* pointer to delay from asat 1 to
                           asat 2 */
             asat2_1,      /* pointer to delay from asat 2 to
                           asat 1 */
             trial_time,   /* pointer to trial time for delay
                           study */
             network_flag, /* indicates network interrupt flag */
             min=0, sec=0; /* trial duration in minutes and
                           seconds */

unsigned long max_sent_1, /* maximum packets transmitted to
                           asat1 */
             max_sent_2, /* maximum packets transmitted to
                           asat2 */
             plus_1,     /* packets sent to asat1 with extra
                           delay */
             plus_2,     /* packets sent to asat2 with extra
                           delay */
             minus_1,    /* packets sent to asat1 with less
                           delay */
             minus_2;    /* packets sent to asat2 with less
                           delay */

double       max_plus_error_1, /* upper range of error in
                           delay to Ast1*/
             max_minus_error_1, /* lower range of error in
                           delay to Ast1*/
             ave_error_1,      /* average error in delay to
                           asat1 */
             max_plus_error_2, /* upper range of error in
                           delay to Ast2*/
             max_minus_error_2, /* lower range of error in
                           delay to Ast2*/
             ave_error_2;      /* average error in delay to
                           asat2 */

```

```
/*  
 * Forwards  
 */
```

```
char alert_window(char *);  
void hit_any(int color,char *buffer,int text);  
void getstr(char buf[],int len,int x,int y);  
void file_data(FILE *fptr);  
void file_header(FILE *fptr);  
void paint(short color,short x1,short y1,short x2,short y2);  
void heading(int color);  
void draw_border(int x1,int y1,int x2,int y2);  
void show(int x,int y,char *buffer);
```

```

/*****
*                               set_values()                               *
*                               *                                           *
*   PURPOSE   :   takes each values from the argument received by *
*                 the program, and assigns them to global          *
*                 variables                                         *
*                               *                                           *
*   ASSUMES   :   nothing                                           *
*                               *                                           *
*   CALL      :   set_values(args);                                  *
*                               *                                           *
*   INPUT PARAMETERS : args -- pointer to character string with*
*                           received parameters                     *
*                               *                                           *
*   FUNCTION CALLS( other than standard functions ) : none        *
*                               *                                           *
*   RETURNS   :   nothing                                           *
*                               *                                           *
*   SIDE EFFECTS : all globals are set                             *
*                               *                                           *
*****/

```

```

void set_values(char *argv[])
{
    /*
       convert each of the parameters from string form
       into int/float/long resp.
    */

    asat1_2 = atoi(argv[1]);
    asat2_1 = atoi(argv[2]);

    trial_time = atoi(argv[3]);

    max_sent_1 = atol(argv[4]);
    max_sent_2 = atol(argv[5]);

    plus_1 = atol(argv[6]);
    plus_2 = atol(argv[7]);

    minus_1 = atol(argv[8]);
    minus_2 = atol(argv[9]);

    max_plus_error_1 = atof(argv[10]);
    max_plus_error_2 = atof(argv[11]);
    max_minus_error_1 = atof(argv[12]);
    max_minus_error_2 = atof(argv[13]);

    ave_error_1 = atof(argv[14]);
    ave_error_2 = atof(argv[15]);

    trial_number = atoi(argv[16]);
    network_flag = atoi(argv[17]);
}

```

```

*****
*                                     done_window()                                     *
*                                                                                       *
* PURPOSE   :   draws the done_window and informs the user that the *
*               trial has been completed *
*                                                                                       *
* ASSUMES   :   nothing *
*                                                                                       *
* CALL      :   done_window(); *
*                                                                                       *
* INPUT PARAMETERS : none *
*                                                                                       *
* FUNCTION CALLS( other than standard functions ) : *
*                                                                                       *
*               paint(color,x1,y1,x2,y2) *
*               heading(color); *
*               draw_border(x1,y1,x2,y2) *
*               hit_any(color,buf,color) *
*               show(x,y,buf); *
*                                                                                       *
* RETURNS   :   nothing *
*                                                                                       *
* SIDE EFFECTS : none *
*****/

```

```

void done_window(void)
{
    _clearscreen(_GCLEARSCREEN);
    paint(WHITE,0,0,700,600);
    heading(GREEN);
    paint(GREEN,140,150,490,250);
    draw_border(140,150,490,250);
    _settextcolor(BRIGHTWHITE);
    _setbkcolor(_GREEN);
    show(13,25," Trial is complete now ");
    hit_any(GREEN, " Hit any key for results ... ",BRIGHTWHITE);
}

```

```

/*****
*                                     result_window()                                     *
*                                     *                                                 *
* PURPOSE : draws the result_window, displays the trial results*
*                                     *                                                 *
* ASSUMES : nothing*
*                                     *                                                 *
* CALL : result_window();*
*                                     *                                                 *
* INPUT PARAMETERS : none*
*                                     *                                                 *
* FUNCTION CALLS( other than standard functions ) :*
*                                     *                                                 *
*                                     paint(color,x1,y1,x2,y2)*
*                                     heading(color);*
*                                     draw_border(x1,y1,x2,y2)*
*                                     hit_any(color,buf,color)*
*                                     show(x,y,buf);*
*                                     *                                                 *
* RETURNS : nothing*
*                                     *                                                 *
* SIDE EFFECTS : none*
*****/

```

```

void result_window(void)
{
    char buffer[20];
    int row = 7, col1 = 5, col2 = 40, col3 = 60;

    min = trial_time / 60; /* convert trial time into mins, secs*/
    sec = trial_time % 60;

    _setvideomode(_VRES16COLOR);
    _clearscreen(_GCLEARSCREEN);
    _wrapon(_GWRAPON);
    paint(WHITE,0,0,700,600);
    heading(BLUE);
    paint(BLUE,15,75,600,400);
    draw_border(15,75,600,400);
    _settextcolor(BRIGHTWHITE);
    _setbkcolor(_BLUE);
}

```



```
/* network_flag indicates the manner in which the trial ended*/  
/* a different message is displayed based on its value */
```

```
switch(network_flag)
```

```
{  
    case 0 :  
    {  
        show(row,5,"Trial termination status : NORMAL");  
        break;  
    }  
  
    case 1 :  
    {  
        show(row,5,"Trial termination status : ASAT 1  
                    CRASHED");  
        break;  
    }  
  
    case 2 :  
    {  
        show(row,5,"Trial termination status : ASAT 2 DEAD");  
        break;  
    }  
  
    case 9 :  
    {  
        show(row,5,"Trial termination status : USER  
                    TERMINATED");  
        break;  
    }  
}
```

```
/* display trial results in two columns */
```

```
row = row + 2;  
show(row,38,"ASAT 1");          /* column headers */  
show(row,58,"ASAT 2");
```

```
row += 1;  
show(row,col1,"Delay induced");  
sprintf(buffer,"%d",asat1_2);  
show(row,col2,buffer);  
sprintf(buffer,"%d",asat2_1);  
show(row,col3,buffer);
```

```
row += 2;  
show(row,col1,"Trial time (min:sec)");  
sprintf(buffer,"%d:%d",min,sec);  
show(row,col2,buffer);  
show(row,col3,buffer);
```

```

row += 2;
show(row,col1,"Packets sent");
sprintf(buffer,"%d",max_sent_1);
show(row,col2,buffer);
sprintf(buffer,"%d",max_sent_2);
show(row,col3,buffer);

row += 2;
show(row,col1,"Packets sent with more delay");
sprintf(buffer,"%d",plus_1);
show(row,col2,buffer);
sprintf(buffer,"%d",plus_2);
show(row,col3,buffer);

row += 2;
show(row,col1,"Packets sent with less delay");
sprintf(buffer,"%d",minus_1);
show(row,col2,buffer);
sprintf(buffer,"%d",minus_2);
show(row,col3,buffer);

row += 2;
show(row,col1,"Upper range of delay error");
sprintf(buffer,"%f",max_plus_error_1);
show(row,col2,buffer);
sprintf(buffer,"%f",max_plus_error_2);
show(row,col3,buffer);

row += 2;
show(row,col1,"Lower range of delay error");
sprintf(buffer,"%f",max_minus_error_1);
show(row,col2,buffer);
sprintf(buffer,"%f",max_minus_error_2);
show(row,col3,buffer);

row += 2;
show(row,col1,"Average delay error");
sprintf(buffer,"%f",ave_error_1);
show(row,col2,buffer);
sprintf(buffer,"%f",ave_error_2);
show(row,col3,buffer);

row += 2;
hit_any(BLUE,"Hit any key to save results ..",BRIGHTWHITE);
}

```

```

/*****
*                               hit_any()                               *
*
*   PURPOSE   :   gets a keyhit from keyboard for continuation      *
*                  purposes                                           *
*
*   ASSUMES   :   nothing                                             *
*
*   CALL      :   hit_any(color,buf,text);                             *
*
*   INPUT PARAMETERS : int color -- background color                 *
*                      text  -- text color                           *
*                      buf   -- text to be displayed in the window*
*
*   FUNCTION CALLS( other than standard functions ) :                *
*
*                      paint(color,x1,y1,x2,y2)                      *
*                      draw_border(x1,y1,x2,y2)                      *
*                      show(x,y,buf);                                 *
*
*   RETURNS   :   nothing                                             *
*
*   SIDE EFFECTS : none                                              *
*****/

```

```

void hit_any(int color,char *buffer,int text)
{
    _displaycursor(_GCURSOROFF);
    _settextcolor(text);
    _paint(color,140,410,490,460);
    _draw_border(140,410,490,460);
    _show(28,25,buffer);

    getch();

    _setvideomode(_DEFAULTMODE);
}

```

```

/*****
*
*                               file_window()
*
* PURPOSE   :   draws the file window, gets a file name from user
*               for saving the file, alerts user if file already
*               exists, gets a new file name or overwrites it based
*               on user's choice, and writes to file
*
* ASSUMES   :   nothing
*
* CALL      :   file_window();
*
* INPUT PARAMETERS : none
*
* FUNCTION CALLS( other than standard functions ) :
*
*               paint(color,x1,y1,x2,y2)
*               draw_border(x1,y1,x2,y2)
*               show(x,y,buf);
*               alert_window(outfile)
*               file_checker(outfile)
*               getstr(buf,value,x,y)
*               hit_any(color,buf, text)
*
* RETURNS   :   nothing
*
* SIDE EFFECTS : none
*****/

```

```

void file_window(void)
{
    char buffer[40],outfile[13];
    FILE *fptr;
    char buf[2],answer;
    int fileexist,gotit = 0,row = 9;

    _setvideomode(_VRES16COLOR);
    _clearscreen(_GCLEARSCREEN);
    paint(WHITE,0,0,700,600);
    heading(GREEN);
    paint(GREEN,50,100,580,330);
    draw_border(50,100,580,330);
    _displaycursor(_GCURSORON);
    _setbkcolor(_GREEN);

```

```

row += 1;
show(row,10,"Do want to save the results of this run : y/n");
_settextposition(row,57);
flushall();
getstr(buf,2,row,57); /* limit answer to 1 character */
answer = buf[0];

while (!gotit)          /* while answer not y or n */
{
    if (tolower(answer) == 'n' || tolower(answer) == 'y')
        gotit = 1;
    else
    {
        /* beep, overwrite with blank, repeat */
        printf("\07");
        _settextposition(row,57);
        show(row,57," ");
        _settextposition(row,57);
        flushall();
        getstr(buf,2,row,57);
        answer = buf[0];
    }
}

if (tolower(answer) == 'n')
{
    /* user does not want to save */
    row = row + 3;
    show(row,10,"Not saving !");
    hit_any(GREEN, "Hit any key to exit ...",BRIGHTWHITE);
}
else
if (tolower(answer))
{
    /* wants to save, get file name */
    row = row + 2;
    flushall();
    show(row,10,"Give file name (no extension, 8 chrs max) :");
    _settextposition(row,57);
    getstr(outfile,9,row,57);          /* get filename with
                                        max 8 chrs */
    strcat(outfile,".LOG\0");          /* add .LOG extension
                                        to it */
    fileexist = file_checker (outfile); /* check if file
                                        exists */
}

```

```

while (fileexist)
{
    /* file exists, alert user, get y or n reply for
    overwriting */
    answer = alert_window(outfile);

    if (tolower(answer) == 'y')
    {
        /* overwrite -- remove file */
        remove(outfile);
        fileexist = 0;
    }
    else
    {
        /* no overwrite -- get new file name, at previous
        row */
        _displaycursor(_GCMOUSEON);
        show(row,10,"");
        show(row,55,"");
        show(row,10,"Give new file name (no extension, 8
        chrs max) :");
        _settextposition(row,57);
        getstr(outfile,9,row,57); /* get new file name,
        8 chrs */
        strcat(outfile,".LOG\0"); /* .LOG extension */
        fileexist = file_checker (outfile); /* check if it
        exists*/
    }
}

/* open the file for writing */
fptr = fopen(outfile,"w");
sprintf(buffer,"The data will be saved in %s", outfile);

row = row +2;
show(row,10,buffer);
row = row + 2;
show(row,10,"Writing to file ....");

file_header(fptr); /* write header data to file */
file_data(fptr); /* write to trial results to file */
fclose(fptr); /* close file */

hit_any(GREEN,"Hit any key to continue ...",BRIGHTWHITE);
}

```

get_str()

PURPOSE : gets a string from the keyboard, for a specified length, checks for its validity, and then returns it to the caller when a Carriage Return key is pressed. The routine allows the user to edit the string while inputting it.

ASSUMES : nothing

CALL : get_str(buf,value,x,y);

INPUT PARAMETERS : int x,y -- where to read input from on screen
int value -- maximum length of string
char buf -- hold the string

FUNCTION CALLS(other than standard functions) :
show(x,y,buf);

RETURNS : nothing

SIDE EFFECTS : none

void getstr(char buf[],int len,int x,int y)

```
{
    char ch, cbuf[2];
    int i,xpos,ypos;

    _settextposition(x,y);
    xpos = x;
    ypos = y;
    i = 0;

    /* while a string is not obtained, repeat */
    do
    {
        ch = getch();          /* read the key pressed */

        if (ch == CARRIAGE)
            buf[i] = '\0';     /* key pressed is Carriage Return */
        else
```

```

{
    if (ch != BACKSPACE) /* key pressed is not BackSpace */
    {
        if (i >= len-1)
            printf("\07"); /* beep */
        else
        {
            /* length not exceeded, save the char */
            _settextposition(xpos,ypos);
            sprintf(cbuf,"%c",ch);
            _outtext(cbuf);
            buf[i] = ch;
            i++;
            ypos++;
        }
    }
    else /* key pressed is BackSpace */
    {
        if (i <= 0)
            printf("\07"); /* beep */
        else
        {
            /* erase previous char on screen, remove it
             from buf, reset cursor position, if length
             is > 0 */
            i--;
            ypos--;
            _settextposition(xpos,ypos);
            show(xpos,ypos," ");
            _settextposition(xpos,ypos);
        }
    }
}
while (i < len && ch != CARRIAGE);

if (i == len-1)
    buf[len-1] = '\0';
}

```



```

/*****
                                alert_window()

```

PURPOSE : draws an alert window, that the named file already exists, gets user's reply, checks validity of reply return after removing the window from screen

ASSUMES : nothing

CALL : alert_window(outfile);

INPUT PARAMETERS : char *outfile -- ptr to filename

FUNCTION CALLS(other than standard functions) :

```

                                paint(color,x1,y1,x2,y2)
                                draw_border(x1,y1,x2,y2)
                                show(x,y,buf);
                                getstr(buf,value,x,y)

```

RETURNS : user's reply of yes or no to overwrite the named file

SIDE EFFECTS : none

```

*****

```

```

char alert_window(char *outfile)
{
    char buffer[60];
    char answer,buf[2];
    int quit = FALSE;

    paint(GREEN,30,380,570,430);
    draw_border(30,380,570,430);
    _settextcolor(30);
    _displaycursor(_G_CURSORON);
    sprintf(buffer," File %s already exists. Overwrite ??
                (y/n)",outfile);
    show(26,10,buffer);
    _settextposition(26,65);
    flushall();
    getstr(buf,2,26,65);    /* limit answer to 1 character */
    answer = buf[0];

```

```

while (!quit)          /* while not right answer provided */
{
    if (tolower(answer) == 'y' || tolower(answer) == 'n')
        quit = TRUE;
    else
    {
        /* beep, overwrite with blank, repeat */
        printf("\7");
        show(26,65," ");
        flushall();
        getstr(buf,2,26,65); /* limit answer to 1 character
*/
        answer = buf[0];
    }
}

_displaycursor(_GCURSOROFF);
_paint(WHITE,30,380,570,430); /* remove current window */
return (answer);
}

```

```

/*****
                                heading()

```

PURPOSE : draws the heading on the screen

ASSUMES : nothing

CALL : heading(color);

INPUT PARAMETERS : int color -- bgcolor for the heading window

FUNCTION CALLS(other than standard functions) :

```

                                paint(color,x1,y1,x2,y2)
                                draw_border(x1,y1,x2,y2)
                                show(x,y,buf);

```

RETURNS : nothing

SIDE EFFECTS : none

```

*****

```

```

void heading(int color)

```

```

{
    char buf[50];

    paint(color,50,10,580,65);
    draw_border(50,10,580,65);
    sprintf(buf,"AVIATION    TRAINER    TECHNOLOGY    LABORATORY");
    _settextcolor(BRIGHTWHITE);
    _setbkcolor(color);
    show(3,18,buf);
}

```

/*
file_checker()

PURPOSE : checks if given file name already exists or not

ASSUMES : nothing

CALL : file_checker(outfile);

INPUT PARAMETERS : char *outfile -- given file name

FUNCTION CALLS(other than standard functions) : none

RETURNS : true or false

SIDE EFFECTS : none

```
int file_checker(char *outfile)
{
    return(fopen(outfile,"r") != NULL);
}
```

```

/*****
                                file header()
*****/

```

PURPOSE : writes header details to the given file

ASSUMES : file is opened

```
CALL      :  file_header(fpr);
```

INPUT PARAMETERS : FILE *fptr -- pointer to opened file

FUNCTION CALLS(other than standard functions) : none

RETURNS : nothing

SIDE EFFECTS : none

```
void file_header(FILE *fptr)
```

{

```
struct tm *ptr;
```

```
time t mytime;
```

```
mytime = time(NULL);
```

```
ptr = localtime (&mytime);      /* get current system time and
                                date */
```

```
fprintf(fptr,"          T R I A L   %d",
        trial number);
```

```
fprintf(fptr, "\n=====");
```

```
fprintf(fptr, "\n\nThe date and time of this trial are :");
```

```
fprintf(fptr,asctime(ptr));
```

```
fprintf(fptr, "\nDelay to ASAT 1 : %d millisecs", asat1 2);
```

```
fprintf(fptr, "\n\nDelay to ASAT 2 : %d millisecs", asat2_1);
```

```
fprintf(fptr, "\n\nTrial duration (min:sec) : %d:%d", min, sec);
```

```

/* network_flag indicates the manner in which the trial ended
*/
/* a different message is displayed based on its value */
switch(network_flag)
{
    case 0 :
    {
        fprintf(fp_ptr, "\n\nTrial termination status : NORMAL");
        break;
    }

    case 1 :
    {
        fprintf(fp_ptr, "\n\nTrial termination status : ASAT 1
        CRASHED");
        break;
    }

    case 2 :
    {
        fprintf(fp_ptr, "\n\nTrial termination status : ASAT 2
        DEAD");
        break;
    }

    case 9 :
    {
        fprintf(fp_ptr, "\n\nTrial termination status : USER
        TERMINATED");
        break;
    }
}

fprintf(fp_ptr,      "\n\n=====
===== \n\n");
}

```

```

/*****
                                file_data()

```

PURPOSE : writes the trial results to the given file

ASSUMES : file is opened

CALL : file_data(ftpr);

INPUT PARAMETERS : FILE *fptr -- pointer to opened file

FUNCTION CALLS(other than standard functions) : none

RETURNS : nothing

SIDE EFFECTS : none

```

****

```

```

void file_data(FILE *fptr)

```

```

{
    fprintf(fptr, "                                ASAT1"
               "                                ASAT2\n\n");

    fprintf(fptr, "Packets sent                                ");
    fprintf(fptr, "\t%d", max_sent_1);
    fprintf(fptr, "\t\t%d\n\n", max_sent_2);

    fprintf(fptr, "Packets sent with more delay ");
    fprintf(fptr, "\t%d", plus_1);
    fprintf(fptr, "\t\t%d\n\n", plus_2);

    fprintf(fptr, "Packets sent with less delay ");
    fprintf(fptr, "\t%d", minus_1);
    fprintf(fptr, "\t\t%d\n\n", minus_2);

    fprintf(fptr, "Upper range of delay error ");
    fprintf(fptr, "\t%f", max_plus_error_1);
    fprintf(fptr, "\t\t%f\n\n", max_plus_error_2);

    fprintf(fptr, "Lower range of delay error ");
    fprintf(fptr, "\t%f", max_minus_error_1);
    fprintf(fptr, "\t\t%f\n\n", max_minus_error_2);

    fprintf(fptr, "Average delay error                                ");
    fprintf(fptr, "\t%f", ave_error_1);
    fprintf(fptr, "\t\t%f\n\n", ave_error_2);

    fprintf(fptr, "=====
               "=====");
}

```

/*****
draw_border()

PURPOSE : draws a double line border, inside a window

ASSUMES : nothing

CALL : draw_border(x1,y1,x2,y2);

INPUT PARAMETERS : int x1,y1,x2,y2 -- left top, right bottom pts

FUNCTION CALLS(other than standard functions) : none

RETURNS : nothing

SIDE EFFECTS : none

void draw_border(int x1,int y1,int x2,int y2)

{
_setcolor(BRIGHTWHITE);

_moveto(x1+5,y1+5);
_lineto(x2-5,y1+5);
_lineto(x2-5,y2-5);
_lineto(x1+5,y2-5);
_lineto(x1+5,y1+5);

_moveto(x1+8,y1+8);
_lineto(x2-8,y1+8);
_lineto(x2-8,y2-8);
_lineto(x1+8,y2-8);
_lineto(x1+8,y1+8);
}


```

/*****
                                show

    PURPOSE   :   writes text, inside a window at x,y

    ASSUMES   :   nothing

    CALL      :   show(x,y,buffer);

    INPUT PARAMETERS : int x,y -- location where to write text
                        char buffer - text to write

    FUNCTION CALLS( other than standard functions ) : none

    RETURNS   :   nothing

    SIDE EFFECTS : none
    *****/

void show(int x,int y,char *buffer)
{
    _settextposition(x,y);
    _outtext(buffer);
}

/*****
                                paint()

    PURPOSE   :   fills a window with given color

    ASSUMES   :   nothing

    CALL      :   paint(color,x1,y1,x2,y2);

    INPUT PARAMETERS : int x1,y1,x2,y2 -- left top, right bottom
                        pts int color

    FUNCTION CALLS( other than standard functions ) : none

    RETURNS   :   nothing

    SIDE EFFECTS : none
    *****/

void paint(short color,short x1,short y1,short x2,short y2)
{
    _setcolor(color);
    _rectangle(_GFillInterior,x1,y1,x2,y2);
}

```

```
/******  
* main driver *  
*****/
```

```
void main(int argc, char *argv[])/* parameters received from  
posdelay.exe */
```

```
{  
    set_values(argv);          /* copy parameters into global  
                                variables */  
    done_window();             /* inform user that the trial is  
                                completed */  
    result_window();           /* display results to user on  
                                screen */  
    file_window();             /* save results to file */  
}
```

```
/* End of file */
```

11.0 Data Collection

The data collection module consists of four files, `3com.h`, `data_col.c`, `netto3l.asm`, and `stamp.asm`. The `3com.h` is a header included by `data_col.c`. `Data_col.c` drives the Ethernet software.

11.1 `3com.h`

Refer to `inet.h` (section 3.1), which closely parallels this header, for more details of the structures defined here. These common structures should be combined in a single header.

```
/* These are structures used only for 3COM board initialization */
/* this structure is for the fake DOS INIT header - this is only used for
   non-driver programs */

/* *** TYPES *** */

struct ini_hdr
{
    char len;
    char non1;
    char non2;
    char non3[2];
    char non4[4];
    char non5[4];
    char non6;
    char cdend[4];
    char *argo;
    short args;
    char non7;
};
```

```

/*-----*/
/* The WhoStruct type is used to store information about */
/* the 3Com adapter. This structure is used internally */
/* by the 3Com device drivers. */
/*-----*/
struct WhoStruct
{
    unsigned char addr[6];           // Our ethernet address
    char ver_major;                  // Major version number of driver
    char ver_minor;                  // Minor version number of driver
    char sub_ver;                    // Subversion of driver
    char type_ds;
    char type_adapter;               // Type of 3Com adapter
    char init_status;
    char reserved;
    char num_tran_buf;
    short size_tran_buf;
    long  ttl_tran_cnt;
    long  ttl_tran_err_cnt;
    long  ttl_tran_timeout_cnt;
    long  ttl_recv_cnt;
    long  ttl_recv_bdr_cnt;
    long  ttl_recv_err_cnt;
    long  ttl_retry_cnt;
    char  xfr_mode;
    char  wait_mode;
    char  hdr_spec_data;
};

/* This structure is used for packet manipulation */
typedef struct
{
    unsigned long  int packet_stamp;
    unsigned short int packet_length;
    unsigned char  data[506];
} Packet;

```

11.2 Data_col.c

```
/******
Description:      This file contains the code which calls the
                  functions provided by the CTO3L.ASM to
                  receive/transmit packets through 3COM
                  EtherLinkii board.

                  ASAT's packets will be read and transformed.
*****

/****** includes *****/
#include <stdio.h>
#include <math.h>
#include <graph.h>
#include <process.h>
#include <conio.h>
#include <time.h>
#include <stddef.h>
#include <dos.h>
#include "3com.h"

/****** constants *****/

#define TRUE      1
#define FALSE     0

#define ASAT1     1
#define ASAT2     2
#define Solitaire1 1
#define Solitaire2 2
#define Team      3
#define Independent 4
#define Allvariables 32
#define Convert    16384.0

#define MaxPacketsInArray 600 /* dependent upon size of structure
                               Packet */
```

```

/***** types *****/
typedef struct
{
    unsigned long    TimeStamp;    /* timestamp variable */
    unsigned long    varSPEED;     /* speed */
    unsigned short   varMFF;       /* missile/flare flag */
    unsigned long    varAltx256;   /* F16-A altitude */
    unsigned char    in_address[6]; /* Ethernet address for the Asat */

    short PLength;    /* Packet length */
    short varWEIGHT;   /* F16A weight */
    short varNAB;      /* afterburner stage */
    short varBRAKE;     /* Brake factor */
    short varGforce;    /* G-force */
    short varCAS;       /* Calibrated airspeed */

    double varMACH;     /* machine number */
    /* NEXT 3 FIELDS MUST BE CONVERTED */

    double varPSI;      /* extra precision heading */
    double varTHT;      /* extra precision climb angle */
    double varPHI;      /* extra precision roll angle */

    double varOXP;      /* X coordinate pitch rate */
    double varOXY;      /* X coordinate yaw rate */
    double varDLP;      /* stick deflection, pitch axis */

    double varDLR;      /* stick deflection, roll axis */
    double varTHPBD;     /* pitch rate */
    double varTHPPBD;    /* pitch angular acceleration */
    double varP;         /* roll angle */
    double varPD;        /* roll angular acceleration */
    double varPX;        /* X coordinate roll rate */
    double varABP;       /* 3rd component of acceleration */

    double varTHRUST;    /* engine thrust value */

    long varX;           /* plane x coordinate */
    long varY;           /* plane y coordinate */
    long varZ;           /* plane z coordinate */
    long varV;           /* F-16A velocity */
    long varVFN;         /* F-16A velocity vector: North */

    long varVFE;         /* F-16A velocity vector: East */

    long varVFU;         /* F-16A velocity vector: Up */
    long varH;           /* plane heading */
    long varC;           /* plane climb */
    long varR;           /* plane roll */

    char varMISSILE;     /* missile structure */
    char varFXP;         /* f-16A status */
    char varPLANEID;     /* plane Id */
}
PackDat;

```

```

/***** externs *****/
extern int count; /* number of pkts in buffer not consumed */
extern char flag; /* indicates pc has received type 10 pkt */

/***** function prototypes and forward declarations *****/
void PackSave(PackDat *Dat, FILE *fp1); /* saves packet data */
void extract(PackDat *Dat, int i, FILE *fp1); /* decodes data
from packet */
void extract1(PackDat *Dat, int i, FILE *fp1); /* decodes data
from packet */
void screenOne(void); /* prints scrn 1
(text part) */
void DisplayScreenOne(PackDat *Dat); /* prints scrn 1
(numeric) */
char choose_asat(void); /* gets asat number */
char CheckBit(int bit, char flag, int c); /* explosion
occurred? */
char Screen(void); /* Displays menu */
void name_prompt(char *ptr, FILE *fp); /* Get file name */

/***** global variables *****/
Packet huge PackArray[MaxPacketsInArray]; /* Array for pkts */
Packet huge *Buffer = &PackArray[0]; /* declare packet
pointer */
struct WhoStruct far *Who;

int ArrayIndex = 0; /* Index to packet array */
int ArrayIndexOut = 0; /* Index to buffer */
int Missilecnt = 0, Missilecnt1 = 0; /* Missile counts for Asat1
& Asat2 */
int Asat1Cnt = 0, Asat2Cnt = 0; /* Packet counts for Asat1
& Asat2 */
int FLAG_ARRAY[Allvariables-1]; /* if flag[i] extract from
pkt */
int counter = 0, counter1 = 0;
int mins, secs, milsecs; /* Variable s for time */
int bit1, bit2, bit3, bit4, val; /* Tells which missile was
fired */

unsigned long cnt = 0; /* total count of packet
extracted */
unsigned long int temp1; /* will hold the starting
time */
unsigned long far *timeptr1; /* timestamp pointer */

/* each ASAT has 4 bits that tells us about the missile status. */
/* The following variables will keep track when the missile has */
/* been fired */

char BitStatus1 = TRUE, BitStatus2 = TRUE, BitStatus3 = TRUE,
BitStatus4 = TRUE;
char BitStatus5 = TRUE, BitStatus6 = TRUE, BitStatus7 = TRUE,

```

```
        BitStatus8 = TRUE;

char StatASAT1 = FALSE; /* Status for ASAT1 in solitaire mode */
char StatASAT2 = FALSE; /* Status for ASAT2 in solitaire mode */
char StatASAT12 = FALSE; /* Status for both in solitaire mode */
char StatASAT3 = FALSE; /* Status for team mode */

/***** packet variables *****/

PackDat ASAT1_dat;
```



```

/*****MAIN ROUTINE *****/

```

```

void main(void)
{
    struct tm *ptr1;          /* DOS time structure pointer */
    time_t Lt;                /* The current time */
    char name[10];            /* Input file name */
    int i;                    /* General index variable */
    int num_left;             /* Track # of packets left */

    PackDat *A1 = &ASAT1_dat; /* ASAT-1 data structure */
    FILE *fpl = 0;            /* Output data file pointer */

    unsigned char             /* Ethernet addresses for ASATs */
        asat1_address[6] = {0x02, 0x60, 0x8c, 0x0d, 0x25, 0xea},
        asat2_address[6] = {0x02, 0x60, 0x8c, 0x0d, 0x20, 0xf5};

    /*
       The header contains information about the name of the
       subject and of the experimenter, as well as the complete
       date. In the case of team mode or solitaire mode (both
       flying) the system will ask you to enter the header
       information twice, once for experimenter #1, and
       experimenter #2.
    */

    char HEADER_ARRAY[6][15];
    char HEADER_ARRAY1[6][15];

    spawnl( P_WAIT, "BE.EXE", "BE", " SA reverse", NULL);

    spawnl( P_WAIT, "BE.EXE", "BE",
        " WINDOW 2,10,4,69 BOLD CYAN ON BLUE SHADOW
        ZOOM", NULL);

    spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 3,21 ' AVIATION RESEARCH LABORATORY' BOLD
        WHITE ON BLUE",
        NULL);

    switch (Screen())
    {
        case Solitaire1:
        {
            StatASAT1 = TRUE; /* Solitaire mode */
            name_prompt(name, fpl); /* Ask the user to enter file
                                     name */
            break;
        }
    }
}

```

```

    case Solitaire2:
    {
        StatASAT2 = TRUE;      /* Solitaire mode */
        name_prompt(name, fp1); /* Ask the user to enter file
                                name */
        break;
    }

    case Team:
    {
        StatASAT12 = TRUE;     /* team mode */
        name_prompt(name, fp1); /* Ask the user to enter file
                                name */
        break;
    }

    case Independent:
    {
        StatASAT3 = TRUE;      /* Both flying in solitaire
                                mode */
        name_prompt(name, fp1); /* Ask the user to enter file
                                name */
        break;
    }
}

/* In all cases we open a file for collected data */
if (StatASAT1 || StatASAT12 || StatASAT2 || StatASAT3)
{
    if ((fp1 = fopen(name, "w")) == NULL)
    {
        printf("\n Cannot open data file\n");
        exit(0);
    }
}

/*
The header contains information about the name of the
subject and of the experimenter, as well as the complete
date. In the case of team mode or solitaire mode (both
flying) the system will ask you to enter the header
information twice, once for experimenter #1, and
experimenter #2.
*/

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " SA reverse",
        NULL);

```

```

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " WINDOW 2,10,4,69 BOLD CYAN ON BLUE SHADOW ZOOM",
        NULL);

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 3,21 'ENTER THE NAME OF THE SUBJECT : "
        "' BOLD WHITE ON BLUE",
        NULL);

gets(HEADER_ARRAY[0]);

if ((StatASAT3) || StatASAT12)
{
    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            " SA reverse",
            NULL);

    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            " WINDOW 2,10,4,69 BOLD CYAN ON BLUE SHADOW ZOOM",
            NULL);

    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            " ROWCOL 3,13 'ENTER THE NAME OF THE SECOND
            SUBJECT : "
            "' BOLD WHITE ON BLUE",
            NULL);

    gets(HEADER_ARRAY1[0]);
}

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " WINDOW 2,10,4,69 BOLD CYAN ON BLUE SHADOW ZOOM",
        NULL);

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 3,21 'ENTER THE NAME OF THE GROUP : ' BOLD
WHITE ON BLUE",
        NULL);

```

```

gets(HEADER_ARRAY[4]);

if (StatASAT3 || StatASAT12)
{
    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            " WINDOW 2,10,4,69 BOLD CYAN ON BLUE SHADOW ZOOM",
            NULL);

    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            " ROWCOL 3,13 'ENTER THE NAME OF THE SECOND GROUP
            : "
            "' BOLD WHITE ON BLUE",
            NULL);

    gets(HEADER_ARRAY1[4]);
}

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " WINDOW 2,10,4,69 BOLD CYAN ON BLUE SHADOW ZOOM",
        NULL);

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 3,21 'ENTER THE NAME OF THE EXPERIMENTER: ' "
        "BOLD WHITE ON BLUE",
        NULL);

gets(HEADER_ARRAY[5]);

if (StatASAT3 || StatASAT12)
{
    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            " WINDOW 2,10,4,69 BOLD CYAN ON BLUE SHADOW ZOOM",
            NULL);

    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            " ROWCOL 3,13 'ENTER THE NAME OF THE SECOND
            EXPERIMENTER: "
            "' BOLD WHITE ON BLUE",
            NULL);

    gets(HEADER_ARRAY1[5]);
}

```

```

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " WINDOW 8,10,10,69 BOLD CYAN ON BLUE SHADOW ZOOM",
        NULL);

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 9,21 '      PRESS ENTER TO START 'BOLD WHITE
          ON BLUE",
        NULL);

getch();      /* gives the user a chance to set up the simulator
               and then start to capture data packets */

Lt = time(NULL);
ptr1 = localtime(&Lt);

if (StatASAT1 || StatASAT2 || StatASAT12 || StatASAT3)
{
    fprintf(fp1,"The date and time of %s experiment is :
              ",HEADER_ARRAY[0]);
    fprintf(fp1,asctime(ptr1));

    /* Sends information in the header to the file pointed to
       by fp1 */

    fprintf(fp1,"\nSubject name #1 :%5s\n",HEADER_ARRAY[0]);

    if (StatASAT3 || StatASAT12)
        fprintf(fp1," \nSubject      name      # 2
                    :%5s\n",HEADER_ARRAY1[0]);

    fprintf(fp1,"\nGroup number #1 :%5s\n",HEADER_ARRAY[4]);

    if (StatASAT3 || StatASAT12)
        fprintf(fp1," \nGroup number#2:%5s\n",HEADER_ARRAY1[4]);

    fprintf(fp1,"\nExperimenter #1 :%5s\n",HEADER_ARRAY[5]);

    if (StatASAT3 || StatASAT12)
        fprintf(fp1," \nExperimenter      # 2
                    :%5s\n\n",HEADER_ARRAY1[5]);

    fprintf(fp1,"The following is the data received from the
                ASAT : \n\n");
}

/* set up data collection screen */

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " SA reverse",
        NULL);

```

```

spawn1( P_WAIT,
        "BE.EXE",
        "BE",
        " WINDOW 2,10,4,69 BOLD CYAN ON BLUE SHADOW ZOOM",
        NULL);

spawn1( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 3,21 ' *****RECORDING DATA*****: ' "
        "BOLD WHITE ON BLUE",
        NULL);

spawn1( P_WAIT,
        "BE.EXE",
        "BE",
        " WINDOW 7,2,22,35 BOLD CYAN ON RED SHADOW ZOOM",
        NULL);

spawn1( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 8,15 'ASAT #2 ' BOLD WHITE ON BLACK",
        NULL);

spawn1( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 9,4 'TimeStamp: ' BOLD WHITE ON RED",
        NULL);

spawn1( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 11,4 'Speed: ' BOLD WHITE ON RED",
        NULL);

spawn1( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 13,4 'Altitude: ' BOLD WHITE ON RED",
        NULL);

spawn1( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 15,4 'Heading: ' BOLD WHITE ON RED",
        NULL);

```

```

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 17,4 'PacketCount: ' BOLD WHITE ON RED",
        NULL);

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 19,4 'PacketLenght: ' BOLD WHITE ON RED",
        NULL);

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 21,4 'MissileCnt: ' BOLD WHITE ON RED",
        NULL);

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " WINDOW 7,42,22,75 BOLD CYAN ON BLUE SHADOW ZOOM",
        NULL);

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 8,55 'ASAT #1 ' BOLD WHITE ON BLACK",
        NULL);

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 9,44 'TimeStamp: ' BOLD WHITE ON BLUE",
        NULL);

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 11,44 'Speed: ' BOLD WHITE ON BLUE",
        NULL);

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 13,44 'Altitude: ' BOLD WHITE ON BLUE",
        NULL);

```

```

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 15,44 'Heading: ' BOLD WHITE ON BLUE",
        NULL);

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 17,44 'PacketCount: ' BOLD WHITE ON BLUE",
        NULL);

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 19,44 'PacketLenght: ' BOLD WHITE ON BLUE",
        NULL);

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 21,44 'MissileCnt: ' BOLD WHITE ON BLUE",
        NULL);

cGettimeptr(&timeptr1); /* get pointer to the clock counter
*/
cGetATimeStamp1();      /* gets the value of the counter
*/
temp1 = *timeptr1;      /* temp1 holds the starting time
*/
init3com();             /* Initialize all network software
*/
set_filter3COM();       /* set receive filter to receive
packets*/

```



```

/* Wait until a key is pressed or we run out of space */
while ( !kbhit() && !flag)
{
    /***** ASAT 1 *****/
    if ((StatASAT1 || StatASAT12 || StatASAT3) && (count > 0))
    {
        /* Make sure that the array is circular */

        if (ArrayIndexOut == MaxPacketsInArray)
            ArrayIndexOut = 0;

        i = ArrayIndexOut;

        /* filter for correct ASAT address */
        /* Asat address is 6 bytes (0-5) and byte 10 for packet
           type */
        if (
            (PackArray[i].data[0] == asat1_address[0]) &&
            (PackArray[i].data[1] == asat1_address[1]) &&
            (PackArray[i].data[2] == asat1_address[2]) &&
            (PackArray[i].data[3] == asat1_address[3]) &&
            (PackArray[i].data[4] == asat1_address[4]) &&
            (PackArray[i].data[5] == asat1_address[5]) &&
            (PackArray[i].data[10] == 8)
        )
        {
            /*-- SAVE VARIABLES OF INTEREST FROM PACKET -- */
            extract(A1,i, fp1); /* Extract variables needed
                                from packet */
            PackSave(A1,fp1); /* Save those variables
                                extracted */
            cnt++; /* increment total packet
                    count */
            Asat1Cnt++; /* count the number of packets
                         received */
            ArrayIndexOut++; /* increment the array index
                              */
            count--; /* decrements after consuming
                     one packet*/
        }
    }
}

```

```

/**** ASAT 2 ****/
if ((StatASAT2 || StatASAT12 || StatASAT3) && (count > 0))
{
    /* Make sure that the array is circular */
    if (ArrayIndexOut == MaxPacketsInArray)
        ArrayIndexOut = 0;

    i = ArrayIndexOut;

    /* filter for correct ASAT address */
    /* Asat address takes 6 bytes [0-5] and byte 10 for
       packet type */
    if(
        (PackArray[i].data[0] == asat2_address[0]) &&
        (PackArray[i].data[1] == asat2_address[1]) &&
        (PackArray[i].data[2] == asat2_address[2]) &&
        (PackArray[i].data[3] == asat2_address[3]) &&
        (PackArray[i].data[4] == asat2_address[4]) &&
        (PackArray[i].data[5] == asat2_address[5]) &&
        (PackArray[i].data[10] == 8)
    )
    {
        /*-- SAVE VARIABLES OF INTEREST FROM PACKET --*/

        extract1(A1,i, fp1);    /* Similar to ASAT 1 */
        PackSave(A1,fp1);
        cnt++;
        Asat2Cnt++;
        ArrayIndexOut++;
        count--;
    }
}

/*
   At this point we might not have exhausted all the
   packets in the array. The variable count will
   tell us how many packets are left in the array
   not yet scanned. Therefore another loop is
   needed.
*/
num_left = count;

```

```

while ((StatASAT1 || StatASAT12 || StatASAT3) && (num_left >
0))
{
    /* filter for correct ASAT address */
    /* Asat address takes 6 bytes [0-5], and byte 10 for packet
    type */

    if (
        (PackArray[i].data[0] == asat1_address[0]) &&
        (PackArray[i].data[1] == asat1_address[1]) &&
        (PackArray[i].data[2] == asat1_address[2]) &&
        (PackArray[i].data[3] == asat1_address[3]) &&
        (PackArray[i].data[4] == asat1_address[4]) &&
        (PackArray[i].data[5] == asat1_address[5]) &&
        (PackArray[i].data[10] == 8)
    )
    {
        extract(A1,i,fp1);
        PackSave(A1,fp1);
        cnt++;
        Asat1Cnt++;
        ArrayIndexOut++;
        num_left--;

        if (ArrayIndexOut == MaxPacketsInArray)
            ArrayIndexOut = 0;

        i = ArrayIndexOut;
    }
}

```

```

while ((StatASAT2 || StatASAT12 || StatASAT3) && (num_left >
0))
{
    /* filter for correct ASAT address */
    /* Asat address takes 6 bytes [0-5] and byte 10 for packet
    type */

    if(
        (PackArray[i].data[0] == asat2_address[0]) &&
        (PackArray[i].data[1] == asat2_address[1]) &&
        (PackArray[i].data[2] == asat2_address[2]) &&
        (PackArray[i].data[3] == asat2_address[3]) &&
        (PackArray[i].data[4] == asat2_address[4]) &&
        (PackArray[i].data[5] == asat2_address[5]) &&
        (PackArray[i].data[10] == 8)
    )
    {
        extract1(A1,i,fp1);
        PackSave(A1,fp1);
        cnt++;
        Asat2Cnt++;
        ArrayIndexOut++;
        num_left--;

        if (ArrayIndexOut == MaxPacketsInArray)
            ArrayIndexOut = 0;

        i = ArrayIndexOut;
    }
}

_clearscreen(_GCLEARSCREEN);

fclose(fp1);      /* close the file they contains the desired
                  data */
reset3COM();      /* reset the 3COM board */

_setbkcolor(0L);  /* set the background color to the initial
                  color */
_setcolor(7);     /* set color to initial color */
}

```

```

/*****
    Displays menu to choose form
*****/

char Screen(void)
{
    char c;

    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            " WINDOW 7,10,22,69 BOLD CYAN ON BLUE SHADOW ZOOM",
            NULL);

    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            " ROWCOL 9,20 '*****DATA COLLECTION APPLICATION*****'
              BOLD "
            "WHITE ON BLUE",
            NULL);

    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            " ROWCOL 11,26 'Collect from A S A T #1.....1' BOLD
              WHITE ON BLUE",
            NULL);

    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            " ROWCOL 13,26 'Collect from A S A T #2.....2' BOLD
              WHITE ON BLUE",
            NULL);

    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            " ROWCOL 15,26 'Team Mode.....3' BOLD
              WHITE ON BLUE",
            NULL);
}

```

```

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 17,26 'Solitaire Mode.....4' BOLD
          WHITE ON BLUE",
        NULL);

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " ROWCOL 20,31 'PRESS A LETTER: ' BOLD CYAN",
        NULL);

c = getch();    /* Give user a chance to read the screen */
while (c < '1' || '5' < c)
    c = getch();

return(c);
}

```

```

/*****
THIS PROMPTS THE USER TO NAME AN ASCII FILE
*****/

```

```

void name_prompt(char *ptr, FILE *fp)
{
    char answer;

    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            "SA reverse",
            NULL);

    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            "WINDOW 11,10,15,69 BOLD CYAN ON BLUE SHADOW ZOOM",
            NULL);

    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            "ROWCOL 12,12 'ENTER FILE NAME:' BOLD WHITE ON BLUE",
            NULL);

    gets(ptr); /* get the file name */

    if (!((fp = fopen (ptr, "r")) == NULL)) /* check if file does
                                            exist */
    {
        /*
        In this case the file does exist. We want to make
        sure to ask the use if we should overwrite the
        file. In the case where the answer is no, the
        user has to enter another file name.
        */

        spawnl( P_WAIT,
                "BE.EXE",
                "BE",
                "SA reverse",
                NULL);
    }
}

```

```

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " WINDOW 17,10,21,69 BOLD CYAN ON BLUE SHADOW
        ZOOM",
        NULL);

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        "ROWCOL 18,12 'FILE ALREADY IN USE' BOLD WHITE ON
        BLUE",
        NULL);

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        "ROWCOL 20,12 'WOULD YOU LIKE TO OVERWRITE THAT
        FILE? ...: "
        "' BOLD WHITE ON BLUE",
        NULL);

do
{
    answer = tolower(getch());
    printf("\n");

    switch (answer)
    {
        case 'n':
        {
            name_prompt(ptr,fp); /* Get a 's file name */
            break;
        }

        case 'y':
        {
            remove(ptr);          /* Overwrite the file */
            break;
        }
    }
}
while (answer != 'n' && answer != 'y');
}

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        "SA reverse",
        NULL);

```



```

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " WINDOW 11,10,15,69 BOLD CYAN ON BLUE SHADOW ZOOM",
        NULL);

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        "ROWCOL 13,12 'The data will be saved into the file: "
        "' BOLD WHITE ON BLUE",
        NULL);

_settextposition( 14,52);
printf("%s",ptr); /* prints the file name where the data will
                  be stored */

getch();
}

```

```

/*****
  THIS IS USED TO GET THE ASAT NUMBER, FROM WHICH PACKETS
  ARE TO BE COLLECTED
*****/

```

```

char choose_asat(void)
{
    printf("Which ASAT do you want to sample from :\n");

    printf(" 1- ASAT1 \n");
    printf(" 2- ASAT2 \n");
    printf(" 3- TEAM MODE \n");

    return(getch());
}

```

```

/*****
make_choice :
    Gives a pick-a-number interface.  It:

    Displays a menu and lets the user enter a choice in the
    form of numbers.  Initialize the display and saving
    procedure for the variables that correspond to the
        n    u    m    b    e    r    s    .
*****/

void make_choice(void)
{
    int VALUE;
    int index;

    memset(FLAG_ARRAY,0,sizeof(FLAG_ARRAY));
    _clearscreen(_GCLEARSCREEN);

    /*
    Print list of variables to choose from in order to extract
    from the packet.
    */

    printf("LIST OF VARIABLES : \n\n");
    printf(" 0- speed, V                                18- vel. north, VFN \n");
    printf(" 1- heading, H                                19- vel. east, VFE \n");
    printf(" 2- climb a., C                                20- vel. up, VFU \n");
    printf(" 3- roll a., R                                21- X position, X \n");
    printf(" 4- mach, MACH                                22- Y position, Y \n");
    printf(" 5- a. psi, PSI                                23- Z position, Z \n");
    printf(" 6- a. theta, THT                             24- altitude, ALT \n");
    printf(" 7- a. phi, PHI                                25- afterburners, NAB \n");
    printf(" 8- p.r. X coord, OXP                          26- thrust, THRUST \n");
    printf(" 9- y.r. Y coord, OXY                          27- brake, BRAKE \n");
    printf("10- r.r. X coord, PX                           28- calib. airspeed, CAS \n");
    printf("11- p.c.s. defl., DLP                          29- gravity, G \n");
    printf("12- r.c.s. defl., DLR                          30- weight, WEIGHT \n");
    printf("13- p. r., THPBD \n");
    printf("14- p.a. acc., THPPBD \n");
    printf("15- roll rate, P \n");
    printf("16- roll a. acc., PD \n");
    printf("17- 3rd cpnent of acc., ABP \n \n");
    printf("    " %d- To choose all the variables
\n    \n    ",
        Allvariables);

    printf("ENTER THE VARIABLES YOU ARE INTERESTED IN SAVING: \n");

    index = 0;

```

```
/* this loop will allow you to take in consideration all the
variables ( when the value Allvariables is entered */
while (scanf("%d",&VALUE) != 0)
{
    if (VALUE == Allvariables)
    {
        for (index = 0; index < (Allvariables-1); index++)
            FLAG_ARRAY[index] = 1;
        break;
    }
    FLAG_ARRAY[VALUE] = 1;
}
}
```

```

/*****
screenone function
this function is used to set up the run-time display screen
*****/

```

```

/* This procedure displays all variables chosen */

```

```

void screenOne(void)
{
    int COUNT, COUNT1; /* COUNT and COUNT1 will indicate line
                        numbers for the display set up */

    make_choice();
    _clearscreen(_GCLEARSCREEN);

    _clearscreen(_GCLEARSCREEN);
    _settextposition (1,1);
    printf("Packet Timestamp: \n \n");

    COUNT = 1;
    COUNT1 = 1;

    if (FLAG_ARRAY[0] == 1)
    {
        COUNT++;
        _settextposition (COUNT,1);
        printf("speed      V:          \n");
    }

    if (FLAG_ARRAY[1] == 1)
    {
        COUNT++;
        _settextposition (COUNT,1);
        printf("heading   H:          \n");
    }

    if (FLAG_ARRAY[2] == 1)
    {
        COUNT++;
        _settextposition (COUNT,1);
        printf("climb a.  C:          \n");
    }
}

```

```

if (FLAG_ARRAY[3] == 1)
{
    COUNT++;
    _settextposition (COUNT,1);
    printf("roll a.   R:           \n");
}

if (FLAG_ARRAY[4] == 1)
{
    COUNT++;
    _settextposition (COUNT,1);
    printf("mach    MACH:           \n");
}

if (FLAG_ARRAY[5] == 1)
{
    COUNT++;
    _settextposition (COUNT,1);
    printf("psi a.   PSI:           \n");
}

if (FLAG_ARRAY[6] == 1)
{
    COUNT++;
    _settextposition (COUNT,1);
    printf("theta a.THT:           \n");
}

if (FLAG_ARRAY[7] == 1)
{
    COUNT++;
    _settextposition (COUNT,1);
    printf("phi a.   PHI:           \n");
}

if (FLAG_ARRAY[8] == 1)
{
    COUNT++;
    _settextposition (COUNT,1);
    printf("pitch r.OXP:           \n");
}

```

```

if (FLAG_ARRAY[9] == 1)
{
    COUNT++;
    _settextposition (COUNT,1);
    printf("yaw r.  OXY:           \n");
}

if (FLAG_ARRAY[10] == 1)
{
    COUNT++;
    _settextposition (COUNT,1);
    printf("roll r.  PX:           \n");
}

if (FLAG_ARRAY[11] == 1)
{
    COUNT++;
    _settextposition (COUNT,1);
    printf("p.c.s.d.DLP:           \n");
}

if (FLAG_ARRAY[12] == 1)
{
    COUNT++;
    _settextposition (COUNT,1);
    printf("r.c.s.  DLR:           \n");
}

if (FLAG_ARRAY[13] == 1)
{
    COUNT++;
    _settextposition (COUNT,1);
    printf("p.r.  THPBD:           \n");
}

if (FLAG_ARRAY[14] == 1)
{
    COUNT++;
    _settextposition (COUNT,1);
    printf("p.a.aTHPPBD:           \n");
}

if (FLAG_ARRAY[15] == 1)
{
    COUNT++;
    _settextposition (COUNT,1);
    printf("roll r.  P:           \n");
}

```

```

if (FLAG_ARRAY[16] == 1)
{
    COUNT++;
    _settextposition (COUNT,1);
    printf("r.a.a.    PD:           \n");
}

if (FLAG_ARRAY[17] == 1)
{
    COUNT++;
    _settextposition (COUNT,1);
    printf("3rd c.a.ABP:           \n");
}

if (FLAG_ARRAY[18] == 1)
{
    COUNT1 ++;
    _settextposition (COUNT1,40);
    printf("vel. north    VFN:           \n");
}

if (FLAG_ARRAY[19] == 1)
{
    COUNT1 ++;
    _settextposition (COUNT1,40);
    printf("vel. east      VFE:           \n");
}

if (FLAG_ARRAY[20] == 1)
{
    COUNT1 ++;
    _settextposition (COUNT1,40);
    printf("vel. up        VFU:           \n");
}

if (FLAG_ARRAY[21] == 1)
{
    COUNT1 ++;
    _settextposition (COUNT1,40);
    printf("X position      X:           \n");
}

if (FLAG_ARRAY[22] == 1)
{
    COUNT1 ++;
    _settextposition (COUNT1,40);
    printf("Y position      Y:           \n");
}

```



```

if (FLAG_ARRAY[23] == 1)
{
    COUNT1 ++;
    _settextposition (COUNT1,40);
    printf("Z position      Z:                \n");
}

if (FLAG_ARRAY[24] == 1)
{
    COUNT1 ++;
    _settextposition (COUNT1,40);
    printf("altitude      ALT:                \n");
}

if (FLAG_ARRAY[25] == 1)
{
    COUNT1 ++;
    _settextposition (COUNT1,40);
    printf("afterburners NAB:                \n");
}

if (FLAG_ARRAY[26] == 1)
{
    COUNT1 ++;
    _settextposition (COUNT1,40);
    printf("thrust      THRUST:                \n");
}

if (FLAG_ARRAY[27] == 1)
{
    COUNT1 ++;
    _settextposition (COUNT1,40);
    printf("brake      BRAKE :                \n");
}

if (FLAG_ARRAY[28] == 1)
{
    COUNT1 ++;
    _settextposition (COUNT1,40);
    printf("calib. airs  CAS:                \n");
}

```

```

if (FLAG_ARRAY[29] == 1)
{
    COUNT1 ++;
    _settextposition (COUNT1,40);
    printf("gravity      G:           \n");
}

if (FLAG_ARRAY[30] == 1)
{
    COUNT1 ++;
    _settextposition (COUNT1,40);
    printf("weight      WEIGHT:       \n");
}
}

```

```

/*****
DisplayScreenOne      :
    Displays to the screen the scrolling values of the
    variables selected by the user.  The values are
    displayed next to the name of the variables
*****/

void DisplayScreenOne(PackDat *Dat)
{
    int COUNT1 = 1;
    int COUNT2 = 1;

    _settextposition(1,25);
    printf("%2d:%2d.%2d" ,mins, secs, milsecs);

    if (FLAG_ARRAY[0] == 1)
    {
        COUNT1 ++;
        /* V */
        _settextposition(COUNT1,25);
        printf("%5.3f ",Dat->varV);
    }
    /*Speed*/
        _settextposition(11,30);    printf("%5.3f",
        Dat->varSPEED);

    if (FLAG_ARRAY[1] == 1)
    {
        COUNT1 ++;
        /* H */
        _settextposition(COUNT1,25);
        printf("%5.3f ",((Dat->varH*90.0) / Convert));
    }

    if (FLAG_ARRAY[2] == 1)
    {
        COUNT1 ++;
        /* C */
        _settextposition(COUNT1,25);
        printf("%5.3f ",(((32767-Dat->varC)*90.0)/ Convert));
    }

    if (FLAG_ARRAY[3] == 1)
    {
        COUNT1 ++;
        /* R */
        _settextposition(COUNT1,25);
        printf("%5.3f ", ((Dat->varR*90.0)/ Convert ));
    }
}

```

```

if (FLAG_ARRAY[4] == 1)
{
    COUNT1 ++;
    /*MACH*/
    _settextposition(COUNT1,25);
    printf("%5.3f ",Dat->varMACH);
}

if (FLAG_ARRAY[5] == 1)
{
    COUNT1 ++;
    /*PSI*/
    _settextposition(COUNT1,25);
    printf("%5.3f ",Dat->varPSI);
}

if (FLAG_ARRAY[6] == 1)
{
    COUNT1 ++;
    /*THT*/
    _settextposition(COUNT1,25);
    printf("%5.3f ",Dat->varTHT);
}

if (FLAG_ARRAY[7] == 1)
{
    COUNT1 ++;
    /*PHI*/
    _settextposition(COUNT1,25);
    printf("%5.3f ",Dat->varPHI);
}

if (FLAG_ARRAY[8] == 1)
{
    COUNT1 ++;
    /*OXP*/
    _settextposition(COUNT1,25);
    printf("%5.3f ",Dat->varOXP);
}

if (FLAG_ARRAY[9] == 1)
{
    COUNT1 ++;
    /*OXY*/
    _settextposition(COUNT1,25);
    printf("%5.3f ",Dat->varOXY);
}

```

```

if (FLAG_ARRAY[10] == 1)
{
    COUNT1 ++;
    /*PX*/
    _settextposition(COUNT1,25);
    printf("%5.3f ",Dat->varPX);
}

if (FLAG_ARRAY[11] == 1)
{
    COUNT1 ++;
    /*DLP*/
    _settextposition(COUNT1,25);
    printf("%5.3f ",Dat->varDLP);
}

if (FLAG_ARRAY[12] == 1)
{
    COUNT1 ++;
    /*DLR*/
    _settextposition(COUNT1,25);
    printf("%5.3f ",Dat->varDLR);
}

if (FLAG_ARRAY[13] == 1)
{
    COUNT1 ++;
    /*THPBD*/
    _settextposition(COUNT1,25);
    printf("%5.3f ",Dat->varTHPBD);
}

if (FLAG_ARRAY[14] == 1)
{
    COUNT1 ++;
    /*THPPBD*/
    _settextposition(COUNT1,25);
    printf("%5.3f ",Dat->varTHPPBD);
}

if (FLAG_ARRAY[15] == 1)
{
    COUNT1 ++;
    /*P*/
    _settextposition(COUNT1,25);
    printf("%5.3f ",Dat->varP);
}

```

```

if (FLAG_ARRAY[16] == 1)
{
    COUNT1 ++;
    /*PD*/
    _settextposition(COUNT1,25);
    printf("%5.3f ",Dat->varPD);
}

if (FLAG_ARRAY[17] == 1)
{
    COUNT1 ++;
    /*ABP*/
    _settextposition(COUNT1,25);
    printf("%5.3f ",Dat->varABP);
}

if (FLAG_ARRAY[18] == 1)
{
    COUNT2++;
    /*VFN*/
    _settextposition(COUNT2,60);
    printf("%5.3f ", Dat->varVFN);
}

if (FLAG_ARRAY[19] == 1)
{
    COUNT2++;
    /*VFE*/
    _settextposition(COUNT2,60);
    printf("%5.3f ", Dat->varVFE);
}

if (FLAG_ARRAY[20] == 1)
{
    COUNT2++;
    /*VFU*/
    _settextposition(COUNT2,60);
    printf("%5.3f ", Dat->varVFU);
}

if (FLAG_ARRAY[21] == 1)
{
    COUNT2++;
    /* X */
    _settextposition(COUNT2,60);
    printf("%5.3f ",(Dat->varX/256.0));
}

```

```

if (FLAG_ARRAY[22] == 1)
{
    COUNT2++;
    /* Y */
    _settextposition(COUNT2,60);
    printf("%5.3f ",(Dat->varY/256.0));
}

if (FLAG_ARRAY[23] == 1)
{
    COUNT2++ ;
    /*Z*/
    _settextposition(COUNT2,60);
    printf("%5.3f ",(Dat->varZ/256.0));
}

if (FLAG_ARRAY[24] == 1)
{
    COUNT2++;
    /*ALT*/
    _settextposition(COUNT2,60);
    printf("%5.3f ",(Dat->varAltX256/256.0));
}

if (FLAG_ARRAY[25] == 1)
{
    COUNT2++;
    /*NAB*/
    _settextposition(COUNT2,60);
    printf("%5.3f ", Dat->varNAB);
}

if (FLAG_ARRAY[26] == 1)
{
    COUNT2++;
    /*THRUST*/
    _settextposition(COUNT2,60);
    printf("%5.3f ", Dat->varTHRUST);
}

if (FLAG_ARRAY[27] == 1)
{
    COUNT2++;
    /*BRAKE*/
    _settextposition(COUNT2,60);
    printf("%d ", Dat->varBRAKE);
}

```

```

if (FLAG_ARRAY[28] == 1)
{
    COUNT2++;
    /*CAS*/
    _settextposition(COUNT2,60);
    printf("%d ", Dat->varCAS);
}

if (FLAG_ARRAY[29] == 1)
{
    COUNT2++;

    #if 0
        DLR _settextposition(18,60);
        printf("%5.3f ", Dat->varDLR);
    #endif

    /*G*/
    _settextposition(COUNT2,60);
    printf("%d ", Dat->varGforce);
}

if (FLAG_ARRAY[30] == 1)
{
    COUNT2++;
    /*WEIGHT*/
    _settextposition(COUNT2,60);
    printf("%d ", Dat->varWEIGHT);
}
}

```



```

/*****
/* extract function : extracts data from a packet */
/*****

/*----- SAVE VARIABLES OF INTEREST FROM PACKET -----*/

void extract1(PackDat *Dat, int i, FILE *fp1)
{
    unsigned long tempVar;

    /* Source ethernet address is in bytes 0 to 5. */
    Dat->in_address[0] = PackArray[i].data[0];
    Dat->in_address[1] = PackArray[i].data[1];
    Dat->in_address[2] = PackArray[i].data[2];
    Dat->in_address[3] = PackArray[i].data[3];
    Dat->in_address[4] = PackArray[i].data[4];
    Dat->in_address[5] = PackArray[i].data[5];
    Dat->TimeStamp = PackArray[i].packet_stamp; /* TimeStamp of
                                                pkt rcvd */
    Dat->PLength = PackArray[i].packet_length; /* packet
                                                length */

    /* transform ticks to mins, secs, milsecs */

    tempVar = ( ((Dat->TimeStamp) - temp1) / 998.53); /* gets
                                                clock ticks */
    mins = (int) (tempVar / 60000);
    secs = (int) (fmod(tempVar, 60000) / 1000);
    milsecs = (int) (fmod(fmod(tempVar, 60000), 1000));

    _settextposition(10,16);

    printf("%2d:%2d.%2d" ,mins, secs, milsecs);

    /* The complete meaning of the following variables is given
       above */
    /* in the ASAT packet structure. */
    /* For example: X is plane x coordinate, Y is plane y
       coordinate, */
    /* Z is plane Z coordinate, H for heading, C for climb and
       ETC.... */

    #if 0
        Dat->varX = *((long far *) &PackArray[i].data[14]); /* X */
        Dat->varY = *((long far *) &PackArray[i].data[18]); /* Y */
        Dat->varZ = *((long far *) &PackArray[i].data[22]); /* Z */
    #endif
}

```

```

Dat->varH = *((unsigned int far *) &PackArray[i].data[26]); /*
      H */
_settextposition(16,14);
printf(" %5.3f ", (float) ((Dat->varH*90.0)/ Convert ));

#if 0
    Dat->varC = *((short far *) &PackArray[i].data[28]); /* C
    Dat->varR = *((short far *) &PackArray[i].data[30]); /* R
#endif

Dat->varSPEED = *((long far *) &PackArray[i].data[32]); /*
      SPEED*/
_settextposition(12,12);
printf("%5.3f", (float) (Dat->varSPEED/256.0));

#if 0
    Dat->varV = *((double far *) &PackArray[i].data[60]);
/* V */
    Dat->varVFN = *((double far *) &PackArray[i].data[60+8]);
/* VFN */
    Dat->varVFE = *((double far *) &PackArray[i].data[60+16]);
/* VFE */
    Dat->varVFU = *((double far *) &PackArray[i].data[60+24]);
/* VFU */
    Dat->varMACH = *((double far *) &PackArray[i].data[60+32]);
/* MACH */
    Dat->varPSI = *((double far *) &PackArray[i].data[60+40]);
/* PSI */
    Dat->varTHT = *((double far *) &PackArray[i].data[60+48]);
/* THT */
    Dat->varPHI = *((double far *) &PackArray[i].data[60+56]);
/* PHI */
    Dat->varOXP = *((double far *) &PackArray[i].data[60+64]);
/* OXP */
    Dat->varOXY = *((double far *) &PackArray[i].data[60+72]);
/* OXY */
    Dat->varPX = *((double far *) &PackArray[i].data[60+128]);
/* PX */
    Dat->varDLP = *((double far *) &PackArray[i].data[60+80]);
/* DLP */
    Dat->varDLR = *((double far *) &PackArray[i].data[60+88]);
/* DLR */
    Dat->varTHPBD = *((double far *) &PackArray[i].data[60+96]);
/* THPBD */
    Dat->varTHPPBD = *((double far *) &PackArray[i].data[60+104]); /* THPPBD */
    Dat->varP = *((double far *) &PackArray[i].data[60+112]);
/* P */
    Dat->varPD = *((double far *) &PackArray[i].data[60+120]);
/* PD */
    Dat->varABP = *((double far *) &PackArray[i].data[60+136]);
/* ABP */
    Dat->varTHRUST = *((double far *) &PackArray[i].data[60+144]); /* THRUST */

```

```

        Dat->varNAP = *((double far *) &PackArray[i].data[60+152]);
/* NAP */
    #endif

    /*ALT*/
    Dat->varAltx256 = *((unsigned long far *)
        &PackArray[i].data[60+160]);
    _settextposition(14,15);
    printf("%5.3f ",( float) (Dat->varAltx256/256.0));

```

```

#if 0
/*CAS*/
Dat->varCAS = *((unsigned short far *)
               &PackArray[i].data[60+164]);

/*NAB*/
Dat->varNAB = *((unsigned short far *)
               &PackArray[i].data[60+166]);

/*BRAKE*/
Dat->varBRAKE = *((unsigned short far *)
                 &PackArray[i].data[60+168]);

/*G*/
Dat->varGforce = *((unsigned short far *)
                  &PackArray[i].data[60+170]);

/*Weight*/
Dat->varWEIGHT = *((unsigned short far *)
                  &PackArray[i].data[60+172]);

#endif

/*missile*/ Dat->varMISSILE = *((char far *)
                             &PackArray[i].data[241]);

bit1 = ((( Dat->varMISSILE) & 0x80) / 80);
bit2 = ((( Dat->varMISSILE) & 0x40) / 40);
bit3 = ((( Dat->varMISSILE) & 0x20) / 20);
bit4 = ((( Dat->varMISSILE) & 0x10) / 10);

_settextposition(18,18);printf("%d",Asat2Cnt);
_settextposition(22,18);printf("%d", Missilecnt1);

/* we have to check the status of the missile for each data
   packet */
BitStatus5 = CheckBit(bit1,BitStatus5,2);
BitStatus6 = CheckBit(bit2,BitStatus6,2);
BitStatus7 = CheckBit(bit3,BitStatus7,2);
BitStatus8 = CheckBit(bit4,BitStatus8,2);

/* the bit 51 will allow us to check if an explosion has
   occurred */
/* ( crash, hit, explosion). If val == 1 then explosion */
Dat->varEXP = *((char far *) &PackArray[i].data[51]);
val = ((( Dat->varEXP) & 0x40) / 40);

/* if the explosion bit is 1 and the flying mode is team then
   one of the ASAT has crashed based on the ethernet address */
if ((val == 1) && (StatASAT12 == TRUE))
{
    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            "SA reverse",
            NULL);
}

```

```

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        " WINDOW 11,10,15,69 BOLD CYAN ON RED SHADOW ZOOM",
        NULL);

spawnl( P_WAIT,
        "BE.EXE",
        "BE",
        "ROWCOL 12,12 '          *****B I N G O *****' "
        "BOLD WHITE ON RED",
        NULL);

while (!(kbhit()))
    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            "ROWCOL 13,12 '          "
            "ASAT #2 IS DEAD'BOLD WHITE ON RED",
            NULL);

reset3COM();
fclose(fp1);
exit(0);
}

_settextposition(20,19);
printf("%d ", Dat->PLength);

/*RealTimeStamp*/
#if 0
    for (j = 0; j < 6; j++)
        Dat->RTStamp[j] = PackArray[i].data[60+174+j];
#endif

/*MFF*/
Dat->varMFF          = *((unsigned short far *)
                        &PackArray[i].data[60+180]);
}

```

```

/*****
/* extract function : extracts data from a packet */
*****/

/*----- SAVE VARIABLES OF INTEREST FROM PACKET -----*/
void extract(PackDat *Dat, int i , FILE *fp1)
{
    unsigned long tempVar;

    Dat->in_address[0] = PackArray[i].data[0];
    Dat->in_address[1] = PackArray[i].data[1];
    Dat->in_address[2] = PackArray[i].data[2];
    Dat->in_address[3] = PackArray[i].data[3];
    Dat->in_address[4] = PackArray[i].data[4];
    Dat->in_address[5] = PackArray[i].data[5];
    Dat->TimeStamp = PackArray[i].packet_stamp;
    Dat->PLength = PackArray[i].packet_length;

    /* Convert ticks into mins, secs, milsecs */
    tempVar = ((Dat->TimeStamp) - temp1) / 998.53;
    mins = (int) (tempVar / 60000);
    secs = (int) (fmod(tempVar, 60000) / 1000);
    milsecs = (int) (fmod(fmod(tempVar, 60000), 1000));

    _settextposition(10,57);
    printf("%2d:%2d.%2d" ,mins, secs, milsecs);

    #if 0
        /* X */ Dat->varX = *((long far *)
    &PackArray[i].data[14]);
        /* Y */ Dat->varY = *((long far *)
    &PackArray[i].data[18]);
        /* Z */ Dat->varZ = *((long far *)
    &PackArray[i].data[22]);
    #endif

    /* H */
    Dat->varH = *((unsigned int far *)
    &PackArray[i].data[26]);
    _settextposition(16,54);
    printf(" %5.3f ",( float) ((Dat->varH*90.0)/ Convert));

    #if 0
        /* C */ Dat->varC = *((short far *)
    &PackArray[i].data[28]);
        /* R */ Dat->varR = *((short far *)
    &PackArray[i].data[30]);
    #endif

    /*SPEED*/
    Dat->varSPEED = *((long far *) &PackArray[i].data[32]);
    _settextposition(12,52);
    printf("%5.3f",(float) (Dat->varSPEED/256.0));

```

```

    #if 0
        /* V */
        Dat->varV      = *((double far *) &PackArray[i].data[60]);
        /*VFN*/
        Dat->varVFN    = *((double far *) &PackArray[i].data[60+8]);
        /*VFE*/
        Dat->varVFE    = *((double far *) &PackArray[i].data[60+16]);
        /*VFU*/
        Dat->varVFU    = *((double far *) &PackArray[i].data[60+24]);
        /*MACH*/
        Dat->varMACH    = *((double far *) &PackArray[i].data[60+32]);
        /*PSI*/
        Dat->varPSI    = *((double far *) &PackArray[i].data[60+40]);
        /*THT*/
        Dat->varTHT    = *((double far *) &PackArray[i].data[60+48]);
        /*PHI*/
        Dat->varPHI    = *((double far *) &PackArray[i].data[60+56]);
        /*OXP*/
        Dat->varOXP    = *((double far *) &PackArray[i].data[60+64]);
        /*OXY*/
        Dat->varOXY    = *((double far *) &PackArray[i].data[60+72]);
        /*PX*/
        Dat->varPX      =      *((double      far      *)
&PackArray[i].data[60+128]);
        /*DLP*/
        Dat->varDLP    = *((double far *) &PackArray[i].data[60+80]);
        /*DLR*/
        Dat->varDLR    = *((double far *) &PackArray[i].data[60+88]);
        /*THPBD */
        Dat->varTHPBD      =      *((double      far      *)
&PackArray[i].data[60+96]);
        /*THPPB*/
        Dat->varTHPPBD      =      *((double      far      *)
&PackArray[i].data[60+104]);
        /* P */
        Dat->varP      =      *((double      far      *)
&PackArray[i].data[60+112]);
        /*PD*/
        Dat->varPD      =      *((double      far      *)
&PackArray[i].data[60+120]);
        /*ABP*/
        Dat->varABP      =      *((double      far      *)
&PackArray[i].data[60+136]);
        /*THRUST*/
        Dat->varTHRUST      =      *((double      far      *)
&PackArray[i].data[60+144]);
        /*NAP*/
        Dat->varNAP      =      *((double      far      *)
&PackArray[i].data[60+152]);
    #endif

    /*ALT*/
    Dat->varAltX256      =      *((unsigned      long      far      *)
&PackArray[i].data[60+160]);
    _settextposition(14,55);
    printf("%5.3f ",( float) (Dat->varAltX256/256.0));

```

```

    #if 0
        /*CAS*/
        Dat->varCAS = *((unsigned short far *)
&PackArray[i].data[60+164]);
        /*NAB*/
        Dat->varNAB = *((unsigned short far *)
&PackArray[i].data[60+166]);
        /*BRAKE*/
        Dat->varBRAKE = *((unsigned short far *)
&PackArray[i].data[60+168]);
        /*G*/
        Dat->varGforce = *((unsigned short far *)
&PackArray[i].data[60+170]);
        /*Weight*/
        Dat->varWEIGHT = *((unsigned short far *)
&PackArray[i].data[60+172]);
    #endif

    /*missile*/
    Dat->varMISSILE = *((char far *) &PackArray[i].data[241]);

    /* The ASAT has four missiles. Track which have been fired */

    bit1 = ((( Dat->varMISSILE) & 0x80) / 80);
    bit2 = ((( Dat->varMISSILE) & 0x40) / 40);
    bit3 = ((( Dat->varMISSILE) & 0x20) / 20);
    bit4 = ((( Dat->varMISSILE) & 0x10) / 10);

    _settextposition(18,57); printf("%d",Asat1Cnt);
    _settextposition(22,57); printf("%d",Missilecnt);

    /* flag carries the value of the returned function, that
decides whether a new missile is launched or an old one is in the
air */

    BitStatus1 = CheckBit(bit1,BitStatus1,1);
    BitStatus2 = CheckBit(bit2,BitStatus2,1);
    BitStatus3 = CheckBit(bit3,BitStatus3,1);
    BitStatus4 = CheckBit(bit4,BitStatus4,1);

    /*Explosion*/
    Dat->varEXP = *((char far *) &PackArray[i].data[51]);

    /* Check the explosion flag. If flag is set, an explosion has
occurred. */
    val = ((( Dat->varEXP) & 0x40) / 40);

```



```

if ((val == 1) && (StatASAT12 == TRUE))
{
    /*
        If the explosion bit is set and Status is team then
        one of the ASATs has crashed based on the ASAT id
    */

    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            "SA reverse",
            NULL);

    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            " WINDOW 11,10,15,69 BOLD CYAN ON RED SHADOW ZOOM",
            NULL);

    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            "ROWCOL 12,12 '          *****B I N G O *****'
"
            "BOLD WHITE ON RED",
            NULL);

    while (!kbhit())
        spawnl( P_WAIT,
                "BE.EXE",
                "BE",
                "ROWCOL 13,12 '          ASAT #1 IS
DEAD'  "
                "BOLD WHITE ON RED",
                NULL);

    reset3COM();
    fclose(fpl);
    exit(0);
}

_settextposition(20,59);
printf("%d ", Dat->PLength);

/*MFF*/
Dat->varMFF = *((unsigned short far *)
&PackArray[i].data[60+180]);
}

```

```

/*****
/* packsave function : saves packet details in a file*/
*****/

void PackSave(PackDat *Dat, FILE *fp1)
{
    if (counter == 0)
    {
        fprintf(fp1, "This file contains the following information:
");
        fprintf(fp1, "\n");
        fprintf(fp1, "Plane Id");
        fprintf(fp1, ", TimesTamp");

        #if 0
            if (FLAG_ARRAY[0] == 1)
        #endif

        fprintf(fp1, ", speed");

        #if 0
            if (FLAG_ARRAY[1] == 1)
        #endif

        fprintf(fp1, ", heading");

        if (FLAG_ARRAY[2] == 1)
            fprintf(fp1, ", climb angle");

        if (FLAG_ARRAY[3] == 1)
            fprintf(fp1, ", roll angle");

        if (FLAG_ARRAY[4] == 1)
            fprintf(fp1, ", mach");

        if (FLAG_ARRAY[5] == 1)
            fprintf(fp1, ", psi");

        if (FLAG_ARRAY[6] == 1)
            fprintf(fp1, ", theta");

        if (FLAG_ARRAY[7] == 1)
            fprintf(fp1, ", phi");

        if (FLAG_ARRAY[8] == 1)
            fprintf(fp1, ", p. r. X coord.");
    }
}

```

```

if (FLAG_ARRAY[9] == 1)
    fprintf(fp1," p. r. Y coord.");

if (FLAG_ARRAY[10] == 1)
    fprintf(fp1," r. r. X coord.");

if (FLAG_ARRAY[11] == 1)
    fprintf(fp1," pitch c. s. defl.");

if (FLAG_ARRAY[12] == 1)
    fprintf(fp1," r. c. s. defl.");

if (FLAG_ARRAY[13] == 1)
    fprintf(fp1," pitch rate");

if (FLAG_ARRAY[14] == 1)
    fprintf(fp1," pith a. acc.");

if (FLAG_ARRAY[15] == 1)
    fprintf(fp1," roll rate");

if (FLAG_ARRAY[16] == 1)
    fprintf(fp1," roll a. acc.");

if (FLAG_ARRAY[17] == 1)
    fprintf(fp1," 3rd cpnent of acc.");

if (FLAG_ARRAY[18] == 1)
    fprintf(fp1," vel. north");

if (FLAG_ARRAY[19] == 1)
    fprintf(fp1," vel. east");

if (FLAG_ARRAY[20] == 1)
    fprintf(fp1," vel. up");

if (FLAG_ARRAY[21] == 1)
    fprintf(fp1," X position");

if (FLAG_ARRAY[22] == 1)
    fprintf(fp1," Y position");

if (FLAG_ARRAY[23] == 1)
    fprintf(fp1," Z position");

#if 0
    if (FLAG_ARRAY[24] == 1)
#endif

```

```

        fprintf(fp1," , altitude");
        if (FLAG_ARRAY[25] == 1)
            fprintf(fp1," , afterburner stage");
        if (FLAG_ARRAY[26] == 1)
            fprintf(fp1," , engine thrust");
        if (FLAG_ARRAY[27] == 1)
            fprintf(fp1," , brake");
        if (FLAG_ARRAY[28] == 1)
            fprintf(fp1," , calib. airspeed");
        if (FLAG_ARRAY[29] == 1)
            fprintf(fp1," , gravity");
        if (FLAG_ARRAY[30] == 1)
            fprintf(fp1," , weight");

        fprintf(fp1, " , Missile Info.");
        fprintf(fp1, " , Missilecnt");
        fprintf(fp1," , PacketLength");
        fprintf(fp1, "\n");

        counter = 1;
    }

    if (Dat->in_address[5] == 0xea )
        fprintf(fp1,"%2x \t", 1);
    else
        fprintf(fp1,"%2x \t", 2);

    fprintf(fp1," %2d:%2d.%3d \t", mins, secs, milsecs);

    #if 0
        if (FLAG_ARRAY[0] == 1)
    #endif

    fprintf(fp1,"%5.3f \t ", ( float) ((Dat->varSPEED) / 256.0));

    #if 0
        if (FLAG_ARRAY[1] == 1)
    #endif

    /* ((Dat->varH*90.0)/Convert));          */
    fprintf(
        fp1,
        " %5.3f \t ",
        (float) ((Dat->varH*90.0)/Convert));

```

```

/* (((32767-Dat->varC)*90.0)/Convert)); */
if (FLAG_ARRAY[2] == 1)
    fprintf(fp1,
            "%5.3f \t ",
            (float) (((32767-Dat->varC)*90.0)/Convert));

/* ((Dat->varR*90.0)/Convert)); */
if (FLAG_ARRAY[3] == 1)
    fprintf(fp1, "      %5.3f      \t      ", (float)
((Dat->varR*90.0)/Convert));

if (FLAG_ARRAY[4] == 1)
    fprintf(fp1, "%5.3f \t ", (float) Dat->varMACH);

if (FLAG_ARRAY[5] == 1)
    fprintf(fp1, "%5.3f \t ", (float) Dat->varPSI);

if (FLAG_ARRAY[6] == 1)
    fprintf(fp1, "%5.3f \t ", (float) Dat->varTHT);

if (FLAG_ARRAY[7] == 1)
    fprintf(fp1, "%5.3f \t ", (float) Dat->varPHI);

if (FLAG_ARRAY[8] == 1)
    fprintf(fp1, "%5.3f \t ", (float) Dat->varOXP);

if (FLAG_ARRAY[9] == 1)
    fprintf(fp1, "%5.3f \t ", (float) Dat->varOXY);

if (FLAG_ARRAY[10] == 1)
    fprintf(fp1, "%5.3f \t ", (float) Dat->varPX);

if (FLAG_ARRAY[11] == 1)
    fprintf(fp1, "%5.3f \t ", (float) Dat->varDLP);

if (FLAG_ARRAY[12] == 1)
    fprintf(fp1, "%5.3f \t ", (float) Dat->varDLR);

if (FLAG_ARRAY[13] == 1)
    fprintf(fp1, "%5.3f \t ", (float) Dat->varTHPBD);

if (FLAG_ARRAY[14] == 1)
    fprintf(fp1, "%5.3f \t ", (float) Dat->varTHPPBD);

if (FLAG_ARRAY[15] == 1)
    fprintf(fp1, "%5.3f \t ", (float) Dat->varP);

if (FLAG_ARRAY[16] == 1)
    fprintf(fp1, "%5.3f \t ", (float) Dat->varPD);

if (FLAG_ARRAY[17] == 1)
    fprintf(fp1, "%5.3f \t ", (float) Dat->varABP);

```

```

if (FLAG_ARRAY[18] == 1)
    fprintf(fp1," %5.3f \t ",(float) Dat->varVFN);

if (FLAG_ARRAY[19] == 1)
    fprintf(fp1," %5.3f \t ",(float) Dat->varVFE);

if (FLAG_ARRAY[20] == 1)
    fprintf(fp1," %5.3f \t ",(float) Dat->varVFU);

if (FLAG_ARRAY[21] == 1)
    fprintf(fp1," %5.3f \t ",(float) (Dat->varX/256.0));

if (FLAG_ARRAY[22] == 1)
    fprintf(fp1," %5.3f \t ",(float) (Dat->varY/256.0));

if (FLAG_ARRAY[23] == 1)
    fprintf(fp1," %5.3f \t ",(float) (Dat->varZ/256.0));

#if 0
    if (FLAG_ARRAY[24] == 1)
#endif

    /* (Dat->varAltx256/256.0)); */
    fprintf(
        fp1,
        "%5.3f \t ",
        (float) (Dat->varAltx256/256.0));

if (FLAG_ARRAY[25] == 1)
    fprintf(fp1,"%5.3f \t ", (float) Dat->varNAB);

if (FLAG_ARRAY[26] == 1)
    fprintf(fp1," %5.3f \t ",(float) Dat->varTHRUST);

if (FLAG_ARRAY[27] == 1)
    fprintf(fp1," %5.3f \t ",(float) Dat->varBRAKE);

if (FLAG_ARRAY[28] == 1)
    fprintf(fp1," %5.3f \t ",(float) Dat->varCAS);

if (FLAG_ARRAY[29] == 1)
    fprintf(fp1,"%d \t ", (int) Dat->varGforce);

if (FLAG_ARRAY[30] == 1)
    fprintf(fp1,"%d \t ", (int) Dat->varWEIGHT);

```

```

fprintf(fp1, "%x ", ((( Dat->varMISSILE) & 0x80) /80));
fprintf(fp1, "%x", ((( Dat->varMISSILE) & 0x40) /40));
fprintf(fp1, "%x", ((( Dat->varMISSILE) & 0x20) /20));
fprintf(fp1, "%x \t", ((( Dat->varMISSILE) & 0x10) /10));

if (Dat->in_address[5] == 0xea )
    fprintf(fp1, "%d \t", Missilecnt);
else
    fprintf(fp1, "%d \t", Missilecnt1);

fprintf(fp1, "%d ", Dat->PLength);
fprintf(fp1, "\n");
}

```

```

/*****Missile Count*****/
Each data packet has a flag for missile fire. Each time when a
missile has been fired the flag will change from 0 to 1 for a
certain period (as long as a missile still in the air). When it
hits something the flag becomes zero. BitStatus will check the
flag change. BitStatus tracks the first change from 0 to 1 in
order to increment the missile count
*****/

```

```

char CheckBit(int bit, char BitStatus, int c)
{
    while ((bit == 1) && BitStatus)
    {
        if (c == 1)
            Missilecnt++;          /* counts missile for ASAT1 */
        else
            Missilecnt1++;         /* counts missile for ASAT2 */

        BitStatus = FALSE;
    }

    if (bit == 0)
        BitStatus = TRUE;

    return(BitStatus);
}

```



```

/***** buffer control *****/
/* Used to update buffer pointer when old buffer is full */
void NextBuffer()
{
    ArrayIndex++;

    if (ArrayIndex == MaxPacketsInArray)
        ArrayIndex = 0;

    Buffer = &PackArray[ArrayIndex];
}

```

11.3 S3com.c

/******

B3COM.C

Description: This file augments the access functions for the 3COM
ETHERNET board provided by TIMETO3L.ASM.

```
#include <stdio.h>
#include <dos.h>
#include "3com.h"
```

```
/******
 *   Constants   *
 *****/
```

```
#define FALSE 0
#define TRUE 1
```

```
/******
 *   Externs     *
 *****/
```

```
extern unsigned long cnt;
```

```
/* variables that indicate status of the ASAT's
   team mode, solitaire mode, or independent)
*/
```

```
extern char          /* ASAT status: */
    StatASAT1,       /* team, solitaire or independent mode */
    StatASAT2,
    StatASAT12,
    StatASAT3;
```

```
extern Packet huge
    *Buffer;          /* Pointer to packets */
```

```
extern unsigned char
    asat_address[6];  /* ethernet address of the ASAT */
```

```
extern struct
    WhoStruct far *Who ; /* Ethernet adapter information */
```

```

/***** global variables *****/
static struct ini_hdr
    parmsdr;

static int
    Adapters = 0;

unsigned long far
    *timeptr;          /* Time for packet captured */

int
    count = 0;         /* Counts the # of packets captured */

char
    flag = FALSE;      /* Stop receiving packets?, time out */

/*****
    this function provides initialization for the 3COM board
*****/

void init3com(void)
{
    char *pointr;
    int rc;
    int rs = 0, i = 0;

    /* initialize 3COM board for network communications */
    /* clear parmsdr structure */

    pointr = (char *) &parmsdr.len;
    for (i=0;i<23;i++) pointr[i] = 0x00;

        /* set parmsdr structure */

    parmsdr.len=0x17;
    parmsdr.argo = "d:\\3com\\31\\ETH523.sys";
    parmsdr.args=getds();
    parmsdr.non7=0x00;

    /*save interrupt vector for future restoration */
    cSavvecs();
    rc=getds();
    rc=cInitParameters(parmsdr);
    rc=cInitAdapters(&Adapters);

    rc=cSetLookAhead(32);

    rc=cWhoAmI(&Who);
    cGettimeptr(&timeptr);
}

```

```

/*****
this function SETS THE 3com BOARD to send data
*****/

void send_filter3COM(void)
{
    int rc, rxf = 0x000c, rrx;
    rc=cWrRxFilter(rxf);
    rc=cRdRxFilter(&rrx);
}

/*****
this function sets the 3COM board receive filter to receive
*****/

void set_filter3COM(void)
{
    int rc, rxf=0x000c, rrx;
    rc=cWrRxFilter(rxf);
    rc=cRdRxFilter(&rrx);
}

/* This function returns a ptr to the ethernet address of the 3COM
adapter */

char far *addr_3COM(void)
{
    return(&Who->addr[0]);
}

/*****
this function resets the 3COM adapter
*****/

void reset3COM(void)
{
    int rc = 0;
    rc=cResetAdapter();
    cFixvecs();
}

```

```

/*****
this function prints some basic statistics for the 3COM board
*****/

```

```

void stats3COM(void)
{
    printf("Total 3COM reception count : %d\n",cnt);

    printf("3COM WhoAmI stats :\n");
    printf("  addr = %02x %02x %02x", Who->addr[0],
    Who->addr[1], Who->addr[2]);
    printf(" %02x %02x %02x\n", Who->addr[3],
    Who->addr[4], Who->addr[5]);
}

```

```

/* Interrupt processing routines required by 3COM package */
/*****
note that these routines are performed inside an interrupt, and
thus have a limited scope - they should be short, and some function
calls (especially ones that use interrupts) will not work well
inside them. Note in particular that PutTxData cannot be called
successfully inside RxProcess, and that FTIME does not work inside
any of them. Note also that for some (similar?) reason,
myExitRcvInt does not work if written in C.
*****/

```

```

void myRxProcess(int Status,int PacketSize,int RequestID,char far
*PacketHeader)
{
    int rc, NumBytes, Flags;
    char far *PacketAddr;

    cGetATimeStamp();
    Buffer->packet_stamp = *timeptr;
    Flags = 0x40;
    NumBytes = PacketSize;
    PacketAddr = &Buffer->data[0];
    rc = cGetRxData(&NumBytes, Flags, RequestID, PacketAddr);
    Buffer->packet_length = PacketSize;

    if (Buffer->data[10] == 10)
        flag = TRUE;
}

```

```

    if (StatASAT12 || StatASAT3)
    {
        /* receive only packets of type 8 from ASAT1 or ASAT2 */
        if ((Buffer->data[10] == 8)
            && ((Buffer->data[5] == 0xea) || (Buffer->data[5] ==
0xf5)))
        {
            NextBuffer(); /* next slot in the array */
            count++;      /* Increment the number of packets
                           encountered*/
        }
    }

    if (StatASAT1)
    {
        /* In the case of solitaire mode receive packets only of
           type 8, and from the designated ASAT */
        if ((Buffer->data[10] == 8) && (Buffer->data[5] == 0xea))
        {
            NextBuffer();
            count++;
        }
    }

    if (StatASAT2)
    {
        /* In the case of solitaire mode receive packets only of
           type 8, and from the designated ASAT */
        if ((Buffer->data[10] == 8) && (Buffer->data[5] == 0xf5))
        {
            NextBuffer();
            count++;
        }
    }
}

#if 0
void myTxProcess(Status, RequestID)
int Status, RequestID;
{
}
#endif

```

11.4 Stamp.asm

title stamp.asm

```
*****
;File: STAMP.ASM
;
;Description: This file contains subroutines which provide a
;              C program with an interface to the 3L 1.0 routines.
;              Based on CTO31.ASM.
;*****
```

```
; Functions called by C
PUBLIC _getstamp
PUBLIC _getclock
PUBLIC _cSavvecs
PUBLIC _cFixvecs
PUBLIC _cGetATimeStamp
PUBLIC _cGetATimeStamp1
```

```
PUBLIC _cGettimeptr
```

```
CODE GROUP _TEXT, DATA, ICODE
```

```
_TEXT segment byte public 'CODE'
DGROUP group _DATA, _BSS
assume cs:_TEXT, ds:DGROUP, ss:DGROUP
_TEXT ends
```

```
DATA segment word public 'CODE'
DATA ends
```

```
ICODE segment word public 'CODE'
ICODE ends
```

```
DATA segment
```

```
-----
;
;
_etext db ?

vectsv dd 22h dup (0) ;save all vectors so we can cleanup
retsav dw ?
-----
;
```

```
DATA ends
```

```

_DATA    segment word public 'DATA'
_de      label    byte
_DATA    ends
_BSS     segment word public 'BSS'
_be      label    byte
_BSS     ends
_DATA    segment word public 'DATA'
_se      label    byte

```

```

_DATA    ends

```

```

_TEXT    SEGMENT
        ASSUME  CS:_TEXT, DS:DGROUP, SS:DGROUP

```

```

;-----
;

```

```

temp_hi      db      0
temp_lo      db      0
temp_hi_bit  db      0
timelo       dw      0
timehi       dw      0

```

```

_getstamp proc near

```

```

        xor     ax,ax
        out     043h,al
        in      al,040h
        mov     cs:temp_lo,al
        in      al,040h
        mov     cs:temp_hi,al
        mov     ah,cs:temp_hi
        mov     al,cs:temp_lo
        ret

```

```

_getstamp endp

```

```

;-----
;

```

```

_getclock proc near                ; get lower two bytes from 0040:006c,
clock data.

```

```

        push ds
        mov     ax,0040h
        mov     ds,ax
        mov     ax,ds:006ch
        pop     ds
        ret

```

```

_getclock endp

```


-----;

This function returns a timestamp constructed of the Timer 0 value and the lowest word of the MS-DOS clock. The Timer 0 is a count-down timer, so it is converted to form a coherent timestamp value. The Timer value is returned in the AX register (low word) and the clock value is returned in the DX register (high word).

```
;
_cGetATimeStamp  proc  near

    push    ds                ;set segment pointer for
                                clock read
    mov     ax,0040h          ;
    mov     ds,ax             ;

    mov     al,0c2h           ;set up for count/status
                                latch

;        cli                ;no ints here
    out     043h,al           ;latch
    mov     dx,ds:006ch       ;get clock lsw
;        sti                ;restore ints

    mov     cs:timehi,dx      ;store time high word

    in      al,040h           ;get status
    and     al,080h           ;get msbit
    mov     cs:temp_hi_bit,al  ;store msbit
    in      al,040h           ;get lsb of count
    mov     cs:temp_lo,al     ;store lsb of count
    in      al,040h           ;get msb of count
    mov     ah,al
    mov     al,cs:temp_lo     ;get count into ax
                                reg
    ror     ax,1
    or      ah,cs:temp_hi_bit ;get back bit 16
    not     ax                ;change from
                                count-down to count-up

    mov     cs:timelo,ax      ;store time low
                                word

    pop     ds                ;restore segment
                                pointer

    ret

_cGetATimeStamp  endp
```

```

_cGetATimeStamp1  proc    near

    push    ds                ;set segment pointer for
                                clock read
    mov     ax,0040h          ;
    mov     ds,ax             ;

    mov     al,0c2h           ;set up for count/status
                                latch

    cli                     ;no ints here
    out     043h,al           ;latch
    mov     dx,ds:006ch       ;get clock lsw
    sti                     ;restore ints

    mov     cs:timehi,dx      ;store time high word

    in      al,040h           ;get status
    and     al,080h           ;get msbit
    mov     cs:temp_hi_bit,al ;store msbit
    in      al,040h           ;get lsb of count
    mov     cs:temp_lo,al     ;store lsb of count
    in      al,040h           ;get msb of count
    mov     ah,al
    mov     al,cs:temp_lo     ;get count into ax
                                reg
    ror     ax,1
    or      ah,cs:temp_hi_bit ;get back bit 16
    not     ax                ;change from
                                count-down to count-up

    mov     cs:timelo,ax      ;store time low
                                word

    pop     ds                ;restore segment
                                pointer

    ret

_cGetATimeStamp1  endp

```

```

;-----
;
; _cSavvecs: This procedure saves the interrupt vector table for
; restoration after the adapter usage is completed.
; The interrupt vector table must be restored BEFORE
; initparameters may be called again, else the 55ms
; RTI handler will (at best) execute an infinite loop.
;
; _cSavvecs returns nothing.
;-----

```

```

_cSavvecs proc      near
    push    ds
    push    es
    push    si
    push    di
    push    cx

    mov     ax,ds
    mov     es,ax
    xor     ax,ax
    mov     ds,ax
    mov     cx,22h*2      ;vectors 0 - 21h, 2 wds per
    mov     di,offset CODE:vectsv
    xor     si,si
    cld
    cli
    rep     movsw          ;save them all
    sti

    pop     cx
    pop     di
    pop     si
    pop     es
    pop     ds
    ret
_cSavvecs endp

```

```

;-----
_cFixvecs:  This routine restores the interrupt table portion
             saved by _Savvecs.

_cFixvecs returns nothing.

;-----
;
_cFixvecs proc      near
    push    es
    push    si
    push    di
    push    cx
    push    ax

    xor     ax,ax
    mov     es,ax
    mov     cx,22h*2      ;vectors 0 - 21h, 2 wds per
    mov     si,offset CODE:vectsv
    xor     di,di
    cld
    cli
    rep     movsw          ;restore them all
    sti

    pop     ax
    pop     cx
    pop     di
    pop     si
    pop     es
    ret
_cFixvecs endp

```

```

;-----
_cGettimeptr  proc  near

    push    bp
    mov     bp,sp
    push    si
    push    ds

    mov     ax,cs

    pop     ds
    mov     si,[bp+4]
    mov     word ptr [si],offset cs:timelo
    mov     word ptr [si+2],ax

    pop     si
    pop     bp
    ret

_cGettimeptr  endp

_TEXT  ends
end

```

11.5 Netto3l.asm

title netto3l.asm

```
;*****  
;  
;File: NETTO3L.ASM  
;  
;Description: This file contains subroutines which provide a  
;              C program interface to the 3L 1.0 routines.  
;              Based on CTO3L.ASM.  
;  
;*****
```

; Functions called from C

```
PUBLIC  _cGetCounter  
PUBLIC  _getds  
PUBLIC  _cInitParameters  
PUBLIC  _cInitAdapters  
PUBLIC  _cResetAdapter  
PUBLIC  _cWhoAmI  
PUBLIC  _cRdRxFilter  
PUBLIC  _cWrRxFilter  
PUBLIC  _cPutTxData  
PUBLIC  _cGetRxData  
PUBLIC  _cSetLookAhead
```

;External C functions

```
;extrn _myExitRcvInt      :near  
extrn _myRxProcess        :near  
extrn _myTxProcess        :near  
extrn recv_pkt_posted     :word  
;extrn packet_data_counter :word  
;extrn _nrecv             :word  
;extrn _nbrds             :word
```

;Functions provide by this file

```
PUBLIC  ExitRcvInt  
PUBLIC  RxProcess  
PUBLIC  TxProcess
```

```

;3L functions
extrn  InitParameters      :near
extrn  InitAdapters       :near
extrn  WhoAmI             :near
extrn  ResetAdapter       :near
extrn  RdRxFilter         :near
extrn  WrRxFilter         :near
extrn  GetRxData          :near
extrn  SetLookAhead       :near
extrn  PutTxData          :near

CODE    GROUP    _TEXT, DATA, ICODE

_TEXT   segment byte public 'CODE'
DGROUP  group _DATA, _BSS
        assume cs:_TEXT, ds:DGROUP, ss:DGROUP
_TEXT   ends

DATA    segment word public 'CODE'
DATA    ends

ICODE   segment word public 'CODE'
ICODE   ends

DATA    segment
his_ds  dw      ?
DATA    ends

_DATA   segment word public 'DATA'
_d@     label   byte

_DATA   ends
_BSS    segment word public 'BSS'
_b@     label   byte
_BSS    ends
_DATA   segment word public 'DATA'
_s@     label   byte

_DATA   ends

_TEXT   SEGMENT
        ASSUME  CS:_TEXT, DS:DGROUP, SS:DGROUP

;-----
;
;
_getds  proc      near
        mov      ax,ds
        mov      cs:his_ds,ax
        ret
_getds  endp

```

;_cInitAdapters: This procedure provides the glue between a C
; program and the 3L 1.0 InitAdapters function.
;

;Calling Sequence:

; int cInitAdapters(&nAdapters)
;

;Input Parameters:

; None
;

;Output Parameters:

; int nAdapters
;

;Returns:

; The return value of the InitAdapters function
;

_cInitAdapters proc near

 push bp

 mov bp,sp

 push si

 push di

 push ds

 mov ax,cs

 mov ds,ax

 mov di,offset CODE:RxProcess

 call InitAdapters

 pop ds

 mov di,word ptr[bp+4]

 mov word ptr[di],cx

 pop di

 pop si

 pop bp

 ret

_cInitAdapters endp


```

;-----
;
;_cInitParameters:  This procedure provides the glue between a C
;                   program and the 3L 1.0 InitAdapters function.
;
;Calling Sequence:
;   int cInitParameters(Parms)
;
;Input Parameters:
;   char *Parms - Pointer to a structure with overrides of default
;               parameters.
;
;Output Parameters:
;   None
;
;Returns:
;   The return value of the InitParameters function
;
;-----
_cInitParameters proc near
    push    bp
    mov     bp,sp
    push    si
    push    di
    push    ds

    mov     bx,[bp+4]
    mov     ax,ds
    mov     es,ax
    mov     ax,cs
    mov     ds,ax

    call    InitParameters

    pop     ds
    pop     di
    pop     si
    pop     bp
    ret
_cInitParameters endp

```

```

;-----
;
;_cResetAdapter:  This procedure provides the glue between a C
;                  program and the 3L 1.0 ResetAdapters function.
;
;Calling Sequence:
;    int cResetAdapter()
;
;Input Parameters:
;    None
;
;Output Parameters:
;    None
;
;Returns:
;    The return value of the ResetAdapter function
;
;-----
_cResetAdapter proc near
    push    bp
    mov     bp,sp
    push    si
    push    di
    push    ds

    mov     dx,0
    mov     ax,cs
    mov     ds,ax

    mov     dl,0
    call    ResetAdapter

    pop     ds
    pop     di
    pop     si
    pop     bp

    ret
_cResetAdapter endp

```

```

;-----
;
;_cWhoAmI:  This procedure provides the glue between a C
;           program and the 3L 1.0 WhoAmI function.
;
;Calling Sequence:
;   int cWhoAmI(&WhoPtr)
;
;Input Parameters:
;   None
;
;Output Parameters:
;   struct WhoStruct far *WhoPtr - Far pointer to the WhoAmI
structure
;
;Returns:
;   The return value of the WhoAmI function
;
;-----
_cWhoAmI proc near
    push    bp
    mov     bp,sp
    push    si
    push    di
    push    ds

    mov     dx,0
    mov     ax,cs
    mov     ds,ax

    call    WhoAmI

    pop     ds
    mov     si,[bp+4]
    mov     Word ptr [si],di
    mov     Word ptr [si+2],es

    pop     di
    pop     si
    pop     bp
    ret
_cWhoAmI endp

```

```

;-----
;
;_cRdRxFilter:   This procedure provides the glue between a C
;                program and the 3L 1.0 RdRxFilter function.
;
;Calling Sequence:
;    int cRdRxFilter(&RxFilter)
;
;Input Parameters:
;    None
;
;Output Parameters:
;    int RxFilter - The receive filter value
;
;Returns:
;    The return value of the RdRxFilter function
;
;-----
_cRdRxFilter proc near
    push    bp
    mov     bp,sp
    push    si
    push    di
    push    ds

    mov     ax,cs
    mov     ds,ax

    mov     dx,0
    call    RdRxFilter

    pop     ds
    mov     di,[bp+4]
    mov     [di],bx

    pop     di
    pop     si
    pop     bp
    ret
_cRdRxFilter endp

```

```

;-----
;
;_cWrRxFilter:   This procedure provides the glue between a C
;                program and the 3L 1.0 WrRxFilter function.
;
;Calling Sequence:
;    int cWrRxFilter(RxFilter)
;
;Input Parameters:
;    int RxFilter - The new receive filter value
;
;Output Parameters:
;    None
;
;Returns:
;    The return value of the WrRxFilter function
;
;-----
_cWrRxFilter proc near
    push    bp
    mov     bp,sp
    push    ds
    push    si
    push    di

    mov     ax,cs
    mov     ds,ax

    mov     dx,0
    mov     ax,[bp+4]
    call    WrRxFilter

    pop     di
    pop     si
    pop     ds
    pop     bp
    ret
_cWrRxFilter endp

```

```

;-----
;
;_cSetLookAhead:   This procedure provides the glue between a C
;                  program and the 3L 1.0 SetLookAhead function.
;
;Calling Sequence:
;    int cSetLookAhead(NumBytes)
;
;Input Parameters:
;    int NumBytes - The number of bytes of look ahead data
;
;Output Parameters:
;    None
;
;Returns:
;    The return value of the SetLookAhead function
;
;-----
_cSetLookAhead proc near
    push    bp
    mov     bp,sp
    push    si
    push    di
    push    ds

    mov     ax,cs
    mov     ds,ax

    mov     dx,0
    mov     ax,[bp+4]
    call    SetLookAhead

    pop     ds
    pop     di
    pop     si
    pop     bp
    ret
_cSetLookAhead endp

```

```

-----
;
;_cPutTxData:   This procedure provides the glue between a C
;               program and the 3L 1.0 PutTxData function.
;
;Calling Sequence:
;   int cPutTxData(TotalPacketLen, NumBytes, Flags, RequestID,
;                 PacketAddr, &NewRequestID)
;
;Input Parameters:
;   int TotalPacketLen - The total packet length (first call only)
;   int NumBytes - The number of bytes to transfer this call
;   int Flags - The DL flags
;   int RequestID - Used if not the first call
;   char far * PacketAddr - A far pointer to the packet
;
;Output Parameters:
;   int NewRequestID - Returned after first call
;
;Returns:
;   The return value of the PutTxData function
;
-----
_cPutTxData proc near
    push    bp
    mov     bp,sp
    push    si
    push    di
    push    ds

    mov     ax,ds
    mov     es,ax

    mov     bx,[bp+4]
    mov     cx,[bp+6]

    mov     dl,byte ptr[bp+8]
    mov     dh,byte ptr[bp+10]
    mov     si,[bp+12]
    mov     di,offset CODE:TxProcess
;   mov     di,0ffffh ; no TxProcess

    call    PutTxData

    pop     ds
    xchg    dh,dl
    xor     dh,dh
    mov     di,[bp+16]
    mov     [di],dx

```

```

        pop    di
        pop    si
        pop    bp
        ret
_cPutTxData endp

;-----
;
;_cGetRxData:   This procedure provides the glue between a C
;               program and the 3L 1.0 GetRxData function.
;
;Calling Sequence:
;   int cGetRxData(&NumBytes, Flags, RequestID, PacketAddr)
;
;Input Parameters:
;   int NumBytes - The number of bytes to transfer this call
;   int Flags - The DL flags
;   int RequestID - The request identifier
;   char far * PacketAddr - A far pointer to the packet to copy
;   the data
;
;Output Parameters:
;   int NumBytes - The actual number of bytes transferred
;
;Returns:
;   The return value of the GetRxData function
;
;-----
_cGetRxData proc near
        push    bp
        mov     bp,sp
        push    si
        push    di
        push    ds

        mov     di,[bp+4]
        mov     cx,ss:[di]
        mov     dl,byte ptr[bp+6]
        mov     dh,byte ptr[bp+8]
        mov     di,[bp+10]
        mov     es,[bp+12]
        call    GetRxData

        pop     ds
        mov     di,[bp+4]
        mov     ss:[di],cx

        pop     di
        pop     si
        pop     bp
        ret
_cGetRxData endp

```



```

;-----
;
;TxProcess:  This procedure is the protocol-side routine which is
;             called when a packet has finished transmitting (see
;             cInitAdapters).  It provides the glue between the 3L
;             1.0 routines and C routine called myTxProcess.
;
;myTxProcess Calling Sequence:
;    void myTxProcess(Status, RequestID)
;
;myTxProcess Input Parameters:
;    int Status - Receive status
;    int RequestID - The request identifier
;
;myTxProcess Returns:
;    Nothing
;
;-----

```

```

TxProcess proc near
    push    bp
    push    si
    push    di
    push    ds
    push    es

    push    ax
    mov     ax,cs:his_ds
    mov     ds,ax
    mov     es,ax
    pop     ax

    xor     cx,cx
    mov     cl,dh
    xor     dh,dh

    push    cx
    push    ax
    call    _myTxProcess

    add     sp,4

    pop     es
    pop     ds
    pop     di
    pop     si
    pop     bp
    ret

```

```

TxProcess endp

```

```

;-----
;ExitRcvInt: This procedure is the protocol-side routine which is
;             called when the 3L has completed a receive interrupt.
;             It provides the glue between the 3L 1.0 routines and
;             C routine called myExitRcvInt.
;
;myExitRcvInt Calling Sequence:
;    void myExitRcvInt()
;
;myExitRcvInt Input Parameters:
;    None
;
;myExitRcvInt Returns:
;    Nothing
;-----
ExitRcvInt proc near
;    push    bp
;    push    ds
;    push    es
;    push    si
;    push    di
;
;    push    ax
;    mov     ax,cs:his_ds
;    mov     ds,ax
;    mov     es,ax
;    pop     ax
;
;    call    _myExitRcvInt
;
;    pop     di
;    pop     si
;    pop     es
;    pop     ds
;    pop     bp
;    iret
ExitRcvInt endp

;_myExitRcvInt proc near
;    ret
;_myExitRcvInt endp

```

```

;-----
;
;RxProcess:  This procedure is the protocol-side routine which is
;             called when a packet has been received (see
;             _cInitAdapters).  It provides the glue between the 3L
;             1.0 routines and C routine called myRxProcess.
;
;myRxProcess Calling Sequence:
;  void myRxProcess(Status, PacketSize, RequestID, PacketHeader)
;
;myRxProcess Input Parameters:
;  int Status - Receive status
;  int PacketSize - Size of the received packet
;  int RequestID - The request identifier
;  char far *PacketHeader - Address of the virtual packet header
;
;myRxProcess Returns:
;  Nothing
;
;-----
RxProcess proc near

```

```

    push    bx
    push    cx
    push    dx
    push    si
    push    di
    push    bp
    push    ds
    push    es
    pushf

    push    es
    push    di

    push    ax
    mov     ax,cs:his_ds
    mov     ds,ax
    mov     es,ax
    pop     ax

    xor     bx,bx
    mov     bl,dh
    xor     dh,dh

    push    bx
    push    cx
    push    ax

```

```

        call    _myRxProcess
        add     sp,10

        popf
        pop     es
        pop     ds
        pop     bp
        pop     di
        pop     si
        pop     dx
        pop     cx
        pop     bx
        ret

```

```

RxProcess endp

```

```

;-----
_cGetCounter proc near

```

```

        push bp
        mov bp,sp
        push si
        push di
        push ds

```

```

        mov ax,cs
        mov ds,ax

```

```

        pop ds
        mov cx,cs:recv_pkt_posted
        ;mov cx,cs:_nrecv
        mov di,word ptr[bp+4]
        mov word ptr[di],cx

```

```

        pop di
        pop si
        pop bp
        ret

```

```

_cGetCounter endp

```

```

_TEXT    ends
end

```

12.0 ASAT.c

Program Cover Sheet

Program Name: asat.c Latest Version: 2.0 Date: 9-15-1991

Languages: C Manufacturer: Microsoft Co. Version: 5.1

Description:

This is one of the program modules comprising the ASAT simulator. It is adapted for the intersimulator purpose of delaying communication between two ASAT simulators. This program receives two parameters from the DOS command line: the PC Ethernet address and the simulator number. The PC Ethernet address is used so the ASAT simulators recognize the acceptable packet sender. The simulator number is used for the ASAT simulator to reject packets which were sent by the same ASAT simulator. That is, a simulator cannot receive packets which it has also sent. The program was modified for the intersimulator delay study.

```
/*
=====
=
=                               ASAT.C
=
=====
*/

#include <stdio.h>

/*
   Main initializes XTAR graphics card and EtherLink II network
   adapter. It then starts the program.
*/

int asat_number;
char far pc_address[6];

void main (int argc, char *argv[])
{
    int i, rc;

    if (argc < 3)
    {
        printf("Command line error ! Type in the following
               sequence: \n");
        printf("Executable file name      PC address      ASAT
               number\n");
        printf("Example: asatxx 02608c0d25ea 1\n");
        exit(0);
    }
}
```

```

/* Convert parameters to integers. All are expected to be HEX */
for (i=0; i<12; i++)
{
    if (argv[1][i] <= '9')
        argv[1][i] = argv[1][i]-'0';
    else
        if (argv[1][i] >= 'a' && argv[1][i] <= 'f')
            argv[1][i] = argv[1][i]-'a'+10;
        else
            argv[1][i] = argv[1][i]-'A'+10;
}

/* First parameter specifies the machine's ethernet address */
for (i=0; i<6; i++)
{
    pc_address[i] = argv[1][i*2]*16+argv[1][i*2+1];
}

asat_number = atoi(argv[2]);
rc = pass_filter(pc_address, asat_number);

init_asatid();

if (rc != asat_number)
{
    printf("Asat ID error, ID = %d    returned ID = %d\n", asat_number, rc);
    exit(0);
}

init_card();           /* initialize XTAR */
initq();
initnetworkflag();     /* initialize network flag, assume
                        it's good */

if (init_dev())
{
    reset_dev();        /* bad network ==> reset adapter */
    setnetworkflag();   /* set network flag to bad */
}

mainasm();             /* program's ready to run */
}

```

12.1 C3ltoc.c

Program Cover Sheet

Program Name: c3ltoc.c Latest Version: 1.0 Date: 9-15-1991

Languages: C Manufacturer: Microsoft Co. Version: 5.1

Description:

This file contains subroutines common to the programs *initsys.c* and *delay.c* as well as prototype subroutines used from the Assembler code. The prototype subroutines are not used in the inter-simulator network programs, but may be used for future development.

/*****

c3ltoc.c : This file contains routines used by various other modules, and so acts as a routine library.

*****/

/*
* includes *
*****/

#include <stdio.h>
#include <dos.h>
#include "inet.h"

/*
* externs *
*****/

extern char far asat1_address[6]; /* Asat simulator addresses */
extern char far asat2_address[6];

extern char mypcaddress[6]; /* local PC address */

```

/*****
*   Functions   *
*****/

/*****
init_parmeter: Initialize the Ether Link II adapter parameters.
                Refer to Ether Link II manual.
Return          : None
*****/

void init_parameter(void)
{
    parmsdr = (struct ini_hdr_t *)
               memset(parmsdr,0,sizeof(ini_hdr));

    parmsdr->len=0x17;
    parmsdr->argo = "c:\\3com\\ether503.sys /A:2e0 /D:1
                  /I:3\\0x0a";
    parmsdr->args=getds();          /* save ds segment */
}

/*****
init_3com_all: Initialize the Ether Link II adapter and the
               receiving buffers.
Return        : None
*****/

void init_3com_all(void)
{
    int return_code, rxf=0x000c, rrx, Adapters=0;

    cGet_Parmsptr(&parmsdr,&TotPktsInQ);
    init_parameter();

    return_code=getds();
    return_code=cInitParameters(parmsdr);
    return_code=cInitAdapters(&Adapters);

    return_code=cSetLookAhead(32);

    return_code=cWhoAmI(&Who);
    return_code=cWrRxFilter(rxf);
    return_code=cRdRxFilter(&rrx);

    return_code=cInitBufPtr();
    cPassHead(&Hdptr);
    cGettimeptr(&timeptr);
    cGetPktrrxPtr(&Pktrrxptr);
}

```



```

/*****
CheckHead: this subroutine is called by 'Rxprocess' to check the
packet address in order to switch the receiving buffer
pointer.

```

```

Return    : 0 - reject packet
           1 - receiving buffer #1
           2 - receiving buffer #2

```

```

****

```

```

int CheckHead(void)
{
    int i,reject,id,type;

    id = Hdptr->inh[8];
    type = Hdptr->inh[10];

    /* Code for one Asat address */

    if (0 < network_mode && network_mode <= 2)
    {
        reject = 0;

        /*
            if network_mode = 1: only receive packets from
            simulator #1 it needs to check the Asat address with
            each packet header or check the simulator ID.
        */

        if (network_mode == ASAT1)
        {
            if (id > 0 && id != ASAT1)
                reject = 1;
            else
            {
                for (i=0; i<6; i++)
                    if (Hdptr->inh[i] != asat1_address[i])
                        reject = 1;
            }

            if (!reject)
                return(1);
            else
                return(0);
        }
    }
}

```

```

/*
    if network_mode = 2: only receive packets from
    simulator #2 it needs to check the Asat address with
    each packet header or check the simulator ID.
*/
if (network_mode == ASAT2)
{
    if (id > 0 && id != ASAT2)
        reject = 1;
    else
    {
        for (i=0; i<6; i++)
            if (Hdptr->inh[i] != asat2_address[i])
                reject = 1;
    }

    if (!reject)
        return(2);
    else
        return(0);
}
else
{
    /*
        Code for two Asat addresses - in team mode. It
        needs to check either the packet addresses or
        simulator IDs to each receiving packet.
    */
    if (id > 0)
    {
        if (id == ASAT1)
        {
            if (network_mode == 99 && type == 9)
                return(0);
            else
                return(1);
        }
        else
        {
            if (id == ASAT2)
                return(2);
            else
                return(0);
        }
    }
    else

```

```

        {
            reject = 0;
            for (i=0; i<6; i++)
                if (Hdptr->inh[i] != asat1_address[i])
                    reject = 1;

            if (!reject)
                return(1);
            else
            {
                reject = 0;
                for (i=0; i<6; i++)
                    if (Hdptr->inh[i] != asat2_address[i])
                        reject = 1;
            }

            if (!reject)
                return(2);
            else
                return(0);
        }
    }

    return(0);
}

/*****
myRxProcess: A high level interrupt subroutine to receive packets.
This subroutine is only a prototype. It can be
developed in a higher level protocol, in 'C' code, to
receive packets.

Not used in 'delay.exe'
*****/

void myRxProcess(int Status,int PacketSize,int RequestID,char far
*PacketHeader)
{
    #if 0
        fprintf(stderr,"Called by ASM - myRxProcess\nNot
            implement yet\n");
        fprintf(stderr,"Status=%d, PacketSize=%d,
            RequestID=%d\n",Status,
            PacketSize,RequestID);
    #endif
}

```

```

/*****
myTxProcess: A high level interrupt subroutine to transmit
              packets. This subroutine is only a prototype. It can
              be developed in a higher level protocol, in 'C' code,
              to transmit packets.

```

Not used in 'delay.exe'

```

*****

```

```

void myTxProcess(int Status,int RequestID)
{
    #if 0
        printf("Called by ASM - myTxProcess\nNot implement
                yet\n");
        printf("Status=%d, RequestID=%d\n",Status, RequestID);
    #endif
}

```

```

/*****
myExitRcvInt: A high level subroutine to exit the interrupt of
               receiving packets.

```

This subroutine is only a prototype. It can be developed in a higher level protocol, in 'C' code, to exit at the end of receiving a packet.

Not used in 'delay.exe'

```

*****

```

```

void myExitRcvInt(void)
{
    #if 0
        printf("Called by ASM - myExitRcvInt\nNot implement
                yet\n");
    #endif
}

```

12.2 Internet.asm

Program Cover Sheet

Program Name: internet.asm Latest Version: 1.1 Date: 9-15-1991

Languages: Assembler Manufacturer: Microsoft Co. Version: 5.1

Description:

This is one of the program modules of the ASAT simulator. It is adapted for the intersimulator communication delay between two ASAT simulators. This program receives data packets only from a specified sender, a PC node, and rejects other packets. It also rejects packets which contain the same simulator number given from the ASAT.c module. After receiving a packet, the original packet address (the address of the other ASAT) is reinstalled in the packet buffer for later use. For packet transmission, the program is responsible for putting the simulator number in each packet to recognize the packet returned later from the PC sender.

INTERNET.ASM

```
=====
#
;
;  ** NOTE: **  To allow this program to terminate cleanly
;  savvecs and fixvecs routines were added to preserve vectors
;  that could be changed.  This allows 3L interrupt hooks to be
;  undone so 3L can be used in an executable program rather than
;  just a permanent driver.
;

                .386

include        ..\include\network.inc
```

```

;define 3L functions
extrn  InitParameters:near
extrn  InitAdapters:near
extrn  WhoAmI:near
extrn  ResetAdapter:near
extrn  RdRxFilter:near
extrn  WrRxFilter:near
extrn  GetRxData:near
extrn  SetLookAhead:near
extrn  PutTxData:near
extrn  KillMigFromNetwork:far

extrn  ethernetaddress:word,send_buffer:byte

public  RxProcess
public  ExitRcvInt

public  _InitQ,HowManyInNetwork
public  CheckNetworkActivity,MasterSlaveDecision
public  PlayerInTeam,SortEthernetAddress,PutPacketBack

public  argstr,networkActivity
public  xmit1,rcvsome,savvecs,fixvecs
public  _init_dev,transmit,_reset_dev

public  _pass_filter
public  _init_asatid

CODE    GROUP    DATA, RCODE

DATA    SEGMENT WORD PUBLIC USE16

;DOS driver init request header format
ini_hd  struc
        db      23          ;hdr len
        db      0
        db      0          ;init cmd
stat     dw      0
        db      8 dup (0)
        db      0          ;num units (not used)
cdend    dd      0          ;code end set here
argo     dw      0          ;arg offset
args     dw      0          ;arg segment
        db      0
ini_hd   ends

```

```

;---- adapter parameter setup string -----
;      this would come from 'device=' on real driver init
argstr db      "bs.sys /A:2e0 /D:1 /I:3",0ah

;---- fake driver init request header for InitParameter input
ih      ini_hd  <,,,,,,offset CODE:argstr,seg CODE,>

      even
vectsv dd      22h dup (0)      ;save all vectors so we can cleanup

id_struct      struc
active         db      0          ;0 = node is not active
count         db      0          ;byte to check packet ID
                                   ;(whether the same packet
                                   ;received twice)
address        db      6 dup(0)   ;ethernet address
planeID        dw      0
headPointer    dw      0
tailPointer    dw      0
pktData        db      Qbytes dup(0)
asatid         dw      0          ;store Asat ID to identify
                                   ;packets
id_struct      ends

; don't change the order
ethernetId     id_struct      MAXNUMBEROFMIGS dup(<>)

ethernetTable  dw      ethernetId
               dw      ethernetId+(size id_struct)
               dw      ethernetId+(size id_struct)*2
               dw      ethernetId+(size id_struct)*3
               dw      ethernetId+(size id_struct)*4
               dw      ethernetId+(size id_struct)*5
               dw      ethernetId+(size id_struct)*6
               dw      ethernetId+(size id_struct)*7
               dw      ethernetId+(size id_struct)*8
               dw      ethernetId+(size id_struct)*9
               dw      ethernetId+(size id_struct)*10

ethernetIndex  dw      ethernetTable
               dw      ethernetTable+2
               dw      ethernetTable+4
               dw      ethernetTable+6
               dw      ethernetTable+8
               dw      ethernetTable+10
               dw      ethernetTable+12
               dw      ethernetTable+14
               dw      ethernetTable+16
               dw      ethernetTable+18
               dw      ethernetTable+20

```

```

networkActivity db      (MAXNUMBEROFMIGS * 2) dup (0)      ;1st byte
                                                              ;= current count
                                                              ;2nd byte
                                                              ;= number of repetition
LENGTH_ACTIVITY equ    $-networkActivity

filter_address db      6 dup (0)                          ; source
address of
                                                              ; filtering
filter_id          dw      0                               ;
asat_id            dw      0                               ;
asat1_address      db      02h,60h,8ch,0dh,25h,0eah        ; Asat #1
                                                              ; address
asat2_address      db      02h,60h,8ch,0dh,20h,0f5h        ; Asat #2
                                                              ; address
seqnumber          dw      0                               ;

DATA      ENDS

```

```

RCODE      SEGMENT WORD PUBLIC USE16
            assume  cs:code, ds:code

```

```

comment #
=====

```

```

_pass_filter Gets filter address and number from 'ASAT.C'
Output: ax = filter_id
=====

```

```

#
_pass_filter      proc      far
                  push      bp
                  mov       bp,sp
                  push      es
                  push      si
                  push      di
                  push      cx
                  push      ds

                  mov       ax, cs
                  mov       es, ax

                  mov       cx,6
                  mov       ds,[bp+8]
                  mov       si,[bp+6]
                  mov       di,offset filter_address
                  rep       movsb
                  mov       ax,cs
                  mov       ds,ax

                  mov       ax,[bp+10]
                  mov       filter_id, ax

```



```

                mov     ax, filter_id

                pop     ds
                pop     cx
                pop     di
                pop     si
                pop     es
                pop     bp
                ret
_pass_filter    endp

```

comment #

=====

_init_asatid Initializes asat ID in ethernetID. It is called by
'ASAT.C'

Calling sequence: init_asatid();

Output: Non

=====

```

#
_init_asatid    proc     far
                push    si
                push    di
                push    dx
                push    cx
                push    bx
                push    ds

                mov     ax, cs
                mov     ds, ax

                mov     di, offset code:ethernetIndex
                mov     bx, offset code:asatl_address
                mov     cx, 0

@initid_cont0:
                inc     cx
                mov     si, [di]                ;pointer to
ethernetTable
                mov     si, [si]                ;pointer to
structure

                cmp     filter_id, cx
                jnz     short @initid_cont1
                inc     cx
                add     bx, 6

```

```

@initid_cont1:      mov     [si].asatid, cx
                    cmp     [si].asatid, 2
                    jg      short @initid_cont2

                    mov     eax, dword ptr [bx]
                    mov     dx, word ptr [bx+4]
                    mov     dword ptr [si].address, eax
                    mov     word ptr [si+4].address, dx

                    add     bx, 6
@initid_cont2:      add     di, 2
                    cmp     cx, MAXNUMBEROFMIGS+1
                    jnz     @initid_cont0

                    pop     ds
                    pop     bx
                    pop     cx
                    pop     dx
                    pop     di
                    pop     si
                    ret
_init_asatid      endp

```

comment #

=====

restore_address Restore transmitter's address in Q, pointed by di.

Input : bx - asatid
 di - points to where the address starts.

Output: di - points to where the address starts.

===== #

```

restore_address proc     near
                    push   si
                    push   di
                    push   cx
                    push   es

                    mov    cx, ds
                    mov    es, cx
                    mov    cx, 6
                    cmp    bx, 1
                    jnz     @restor1
                    mov    si, offset asat1_address
                    jmp     short @restor2

```

```

@restor1:      mov     si, offset asat2_address
@restor2:      rep     movsb

                pop     es
                pop     cx
                pop     di
                pop     si
                ret
restore_address endp

```

comment #

```

=====
_Init_Dev initializes network adapter.
Output: ax = 0 (successful)
=====

```

```

#
_init_dev      proc     far

                push    ds
                push    es

                mov     ax, cs
                mov     ds, ax
                mov     es, ax

                call    savvecs

                mov     bx, offset code:ih
                call    InitParameters
                or      ax, ax
                jnz     short @id_exit

                mov     di, offset code:RxProcess
                call    InitAdapters
                or      ax, ax
                jnz     short @id_exit

                xor     dx, dx
                call    WhoAmI
                or      ax, ax
                jnz     short @id_exit

```

```

        mov     si, di
        mov     di, es
        mov     ds, di
        mov     di, seg ethernetaddress
        mov     es, di
        mov     di, offset ethernetaddress
        movsd
        movsw

        mov     dx, cs
        mov     ds, dx

        xor     dx, dx
        mov     cx, 16
        call    SetLookAhead
        or      ax, ax
        jnz     short @id_exit

        xor     dx, dx
        mov     ax, 0ch
        call    WrRxFilter
        or      ax, ax
        jnz     short @id_exit
        pop     es
        pop     ds
        ret

@id_exit:
        push    ax
        call    ResetAdapter
        call    fixvecs
        pop     ax
        pop     es
        pop     ds
        ret

_init_dev    endp

comment #
=====
Reset_Dev resets and reinitializes network adapter.
=====
#
_reset_dev  proc    far

        push    ds
        push    es
        mov     ax, cs
        mov     ds, ax
        mov     es, ax

        call    ResetAdapter
        call    fixvecs

        pop     es
        pop     ds

```

```

        ret
_reset_dev endp

```

```
comment #
```

```
=====
Transmit prepares parameter to transmit a packet.
```

```
Input: bx = length
```

```
       es:[si] = buffer
```

```
Output: ax = error code (0 = successful)
```

```
=====
#
transmit      proc      far
;
              push      ds
              push      es
              mov        ax, cs
              mov        ds, ax
              call       xmit1
              pop        es
              pop        ds
              ret
transmit      endp

```

```
comment #
```

```
=====
Xmit1 passes control to 3L routine in transmit process.
```

```
=====
#
xmit1      proc      near
              mov        dx, 50h
              mov        cx, bx
              mov        di, 0ffffh      ;no TxProcess
              push       es
              pop        ds
              push       si
              push       ds
              push       cx

              push       eax              ;
;      inc      cs:seqnumber              ;
;      mov      ax, cs:seqnumber          ;
              mov        ax, cs:filter_id ;
              mov        word ptr es:[si+8], ax      ;install filter ID

              push       ds              ;
              push       di              ;

              mov        di, seg asat1_address ;
              mov        ds, di          ;
              cmp        ax, 1           ;
              jnz        short @addr1    ;
              mov        di, offset asat1_address ;
              jmp        short @addr2

```

```

@addr1:      cmp     ax, 2                ;
              jnz     short @addr3        ;
              mov     di, offset asat2_address ;
;            mov     di, seg ethernetaddress ;
;            mov     ds, di              ;
;            mov     di, offset ethernetaddress ;
@addr2:      mov     eax, dword ptr ds:[di] ;
              mov     dword ptr es:[si], eax ;
              mov     ax, word ptr ds:[di+4] ;
              mov     word ptr es:[si+4], ax ;
@addr3:      pop     di                  ;
              pop     ds                  ;
              pop     eax                 ;
              call    PutTxData

              pop     cx
              pop     ds
              pop     si
              or      ax, ax
              jz      short xmit1_exit

xmit1_cont0:  push    si
              push    ds
              push    cx
              push    dx
              xor     dl, dl

```

```

        push    eax
        mov     ax, cs:filter_id
        mov     word ptr es:[si+8], ax
        pop     eax
        call    PutTxData
        ;
        ;install filter ID
        ;

        pop     dx
        pop     cx
        pop     ds
        pop     si
        or      ax, ax
        jnz     short xmit1_cont0
xmit1_exit:
        ret
xmit1     endp

```

comment #

```

=====

RcvSome gets a packet from queue buffer.
Inputs: ax = which mig
Returns: ax = 0 - no error
        cx = old head pointer
        ds:[si] = buffer

```

```

=====
#
rcvsome      proc      far
;
        mov     si, cs
        mov     ds, si

        shl     ax, 1
        mov     si, offset code:ethernetIndex
        add     si, ax
        mov     si, [si]           ;pointer to ethernetTable
        mov     si, [si]           ;pointer to structure
        mov     ax, [si].headPointer
        cmp     ax, [si].tailPointer
        jz      short @rcv_exit

        mov     bx, ax
        mov     cx, ax
        inc     bx
        cmp     bx, MaxRecords
        jl      short @rcv_cont0
        xor     bx, bx

```

```

@rcv_cont0:      mov     [si].headPointer, bx

                  mov     bx, [si].planeID
                  imul    ax, PacketLength
                  lea     si, [si].pktData
                  add     si, ax
                  xor     ax, ax
                  ret

@rcv_exit:        mov     bx, [si].planeID
                  mov     ax, -1
                  ret

rcvsome           endp

```

comment ~

```

=====

PutPacketBack puts a packet back in the queue.
Input: ax = which mig
       cx = old head pointer
Output: packet is put back in the queue

=====

```

```

PutPacketBack    proc     far
;
                  push    ds
                  mov     dx, cs
                  mov     ds, dx

                  shl     ax, 1
                  add     ax, offset code:ethernetIndex
                  mov     si, ax
                  mov     si, [si]
                  mov     si, [si]
                  mov     [si].headPointer, cx

                  pop     ds
                  ret

PutPacketBack    endp

```


comment #

=====

RxProcess gets called from either an interrupt handler responding to a packet received interrupt or a subroutine of polling control module which has noted a received packet.

Input: ax = receive status (0 means good, 1 means bad)
cx = size of received packet
dh = request ID
es:di = address of virtual packet header (see SetLookAhead)
currently is not used (this will tell if we want the packet)

=====

```
#
RxProcess      proc      near
;
                pushad
                push      ds
                push      es

                or        ah, ah
                jz         short @rxp_cont9
                xor        cx, cx                ;release buffer
;               jmp        short @rxp_cont4
                jmp        @rxp_cont4           ;

@rxp_cont9:
                cmp        cx, PacketLength
                ja         short @rxp_cont10

                mov        eax, es:[di]        ;get transmitter's address
                mov        bx, es:[di+4]       ;from the packet header

                push       ecx                  ;
                push       dx                  ;

                mov        ecx, dword ptr filter_address ;
                mov        dx, word ptr filter_address+4 ;
                cmp        ecx, eax            ; compare addresses
                jnz        @rxp_cont21         ;
                cmp        dx, bx              ;      "      "
                jnz        @rxp_cont21         ;
                mov        dx, es:[di+8]       ; get simulator ID
                mov        asat_id, dx         ; save simulator ID
                cmp        dx, filter_id       ; compare simulator ID
                jz         @rxp_cont21         ;
```

```

        pop      dx                ;
        pop      ecx              ;
@rxp_cont21:  jmp      @rxp_cont22  ;
        pop      dx                ;
        pop      ecx              ;
buffer      xor      cx, cx        ; release
        jmp      short @rxp_cont4  ;

@rxp_cont22:                                     ;

        mov      di, cs
        mov      ds, di
        mov      es, di

        mov      di, offset code:ethernetIndex
        mov      bp, MAXNUMBEROFMIGS

@rxp_cont0:  mov      si, [di]      ;pointer to ethernetTable
        mov      si, [si]      ;pointer to structure
        cmp      [si].active, 0  ;active?
        jz       short @rxp_cont1 ;no, insert

        push     ax                ;
        mov      ax, [si].asatid  ;
        cmp      asat_id, ax      ;compare asat ID
        pop      ax                ;

;        cmp      eax, dword ptr [si].address
;        jnz      short @rxp_cont2 ;get next one

;        cmp      bx, word ptr [si+4].address
;        jz       short @rxp_cont3 ;found

@rxp_cont2:  add      di, 2
        dec      bp
        jnz      short @rxp_cont0

@rxp_cont10: xor      cx, cx        ;release buffer
        jmp      short @rxp_cont4

@rxp_cont1:  not      [si].active  ;make it
active
;        mov      dword ptr [si].address, eax ;save
ethernet
;        mov      word ptr [si+4].address, bx ;address
;        xor      ax, ax
        jmp      short @rxp_cont5

```

```

@rxp_cont3:      inc      [si].count      ;increment counter
                  mov      ax, [si].tailPointer
                  imul     ax, PacketLength

@rxp_cont5:      lea      di, [si].pktData
                  add      di, ax

@rxp_cont4:      push     di              ;
                  push     si              ;
                  push     cx              ;save size of packet
                  push     ds              ;
                  mov      dl, 040h        ;release buffer
                  call     GetRxData
                  or       ax, ax
                  pop      ds
                  pop      ax
                  pop      si
                  pop      bx              ;bx = di
                  jnz      short @rxp_cont6 ;fail
                  jcxz     short @rxp_cont6 ;fail

                  cmp      ax, cx
                  jnz      short @rxp_cont6 ;fail

                  mov      di, bx          ;di = pointer in [si].pktData
                  mov      bx, [si].asatid ;
                  call     restore_address ;restore
                                          ;transmitter's address
                                          ;pointed by di

                  mov      ax, [si].tailPointer
                  inc      ax
                  cmp      ax, MaxRecords
                  jl       short @rxp_cont7
                  xor      ax, ax

@rxp_cont7:      mov      [si].tailPointer, ax
                  mov      bx, [si].headPointer
                  cmp      ax, bx
                  jnz      short @rxp_cont6
                  inc      bx
                  cmp      bx, MaxRecords
                  jl       short @rxp_cont8
                  xor      bx, bx

@rxp_cont8:      mov      [si].headPointer, bx

@rxp_cont6:      pop      es
                  pop      ds
                  popad
                  ret

RxProcess      endp

```

```

;-----
; _InitQ resets Q pointers.
;-----
;
_InitQ      proc      far
;
;          push      es
;          mov       ax, seg ethernetId
;          mov       es, ax
;          mov       di, offset ethernetId
;          mov       cx, MAXNUMBEROFMIGS
;          xor       eax, eax
;          cli                               ;disable interrupts
@iq_cont0:
;          stosw
;          add       di, 6                    ;skip address
;          stosd
;          stosd
;          stosd
;          stosw
;          add       di, Qbytes
;          add       di, Qbytes+2            ;
;          loop      @iq_cont0
;          sti                               ;turn interrupts back on
;
;          mov       di, offset networkActivity
;          mov       cx, LENGTH_ACTIVITY
;          xor       al, al
;          rep       stosb
;
;          pop       es
;          ret
_InitQ      endp
;-----
;
; HowManyInNetwork calculates number of actives nodes in the
; network.
; Input: nothing
; Output: ax = number of active nodes
;-----
;
HowManyinNetwork      proc      far
;
;          push      ds
;          mov       ax, cs
;          mov       ds, ax

```

```

                                mov     cx, MAXNUMBEROFMIGS
                                mov     si, offset code:ethernetIndex
                                xor     ax, ax
@hmin_cont0:
                                mov     di, [si]           ;pointer to ethernetTable
                                mov     di, [di]           ;pointer to structure
                                cmp     [di].active, ah
                                jz      short @hmin_cont1
                                inc     ax
@hmin_cont1:
                                add     si, 2
                                loop    @hmin_cont0
                                pop     ds
                                ret
HowManyInNetwork               endp

```

comment -

=====

CheckNetworkActivity checks if all the nodes in the network are still active. If one has become inactive for more than 32 frames. that node is assumed to be dead.

Input: none

Output: ax = number of active nodes

=====

```

CheckNetworkActivity          proc     far
;
                                push    ds
                                mov     ax, cs
                                mov     ds, ax

                                mov     cx, MAXNUMBEROFMIGS
                                mov     si, offset code:ethernetIndex
                                mov     bx, offset code:networkActivity
                                xor     dx, dx
                                xor     bp, bp
@cna_cont0:
                                mov     di, [si]           ;pointer to ethernetTable
                                mov     di, [di]           ;pointer to structure
                                cmp     [di].active, dl     ;is it active?
                                jz      short @cna_cont3     ;exit

                                mov     al, [di].count      ;get counter
                                cmp     al, [bx]            ;is it the same one?
                                jnz     short @cna_cont2     ;no, it is a new
                                                        ;packet

```

```

        inc      byte ptr [bx+1]      ;update repetition
                                           ;counter
        cmp      byte ptr [bx+1], 64  ;if there is no
                                           ;activity
                                           ;for 64 frames, kill
                                           ;him

        jb       short @cna_cont1

        mov      word ptr [di].active, dx
        mov      [di].tailPointer, dx
        mov      [di].headPointer, dx

        cmp      cx, 1
        jz       short @cna_cont3

        push     di
        mov      di, [si+2]           ;pointer to ethernetTable
        mov      di, [di]             ;pointer to structure
        cmp      [di].active, dl      ;is next one active?
        pop      di
        jz       short @cna_cont4      ;exit

        mov      ax, [di].planeID
        call     far ptr KillMigFromNetwork

        call     AdjustPointer        ;restructure ethernetIndex
        dec      cx
        jz       short @cna_cont3
        jmp      short @cna_cont0

@cna_cont2:
        mov      [bx], al              ;save packet count
        mov      [bx+1], dl           ;zero out wait counter

@cna_cont1:
        inc      bp
        add      bx, 2
        add      si, 2
        loop     @cna_cont0

@cna_cont3:
        mov      ax, bp
        pop      ds
        ret

@cna_cont4:
        mov      ax, [di].planeID
        call     far ptr KillMigFromNetwork
        mov      ax, bp
        pop      ds
        ret

CheckNetworkActivity endp

```

comment -

=====

AdjustPointer restructures ethernetIndex and counter table if a node has not been transmitting anything for more than 32 frames.

Input: si = current pointer to ethernetIndex

cx = current value of loop counter

bx = current pointer to networkActivity

Output: ethernetIndex and networkActivity have been reorganized.

=====

```
AdjustPointer  proc    near
;
                push    es
                push    si

                mov     ax, ds
                mov     es, ax

                mov     di, si
                push    di
                push    cx

                add     si, 2
                dec     cx

                rep     movsw
                pop     cx
                pop     di
                mov     ax, [di]
                stosw

                push    cx
                mov     di, bx
                mov     si, bx
                add     si, 2
                dec     cx
                rep     movsw
                mov     [di], cx

                pop     cx
                pop     si
                pop     es
                ret
```

@ap_cont0:

```
pop    cx
pop    di
pop    si
pop    es
ret
```

AdjustPointer endp

comment -

=====

MasterSlaveDecision finds out which one the master is by finding the smallest Ethernet address.

Input: ax = number of players-1

Output: ax = ID #

0 - master

non zero positive - slave

-1 - error

=====

-

MasterSlaveDecision

proc far

;

push ds

mov bx, seg ethernetAddress

mov ds, bx

mov bx, offset ethernetAddress

mov edx, [bx]

mov bp, [bx+4]

mov bx, cs

mov ds, bx

push ax

call far ptr HowManyInNetwork

pop bx

mov di, -1

cmp ax, bx

jb short @msd_cont2 ;wait for some
;more people

mov cx, ax

mov si, offset code:ethernetIndex

xor di, di


```

@msd_cont0:
        lodsw
        mov     bx, ax           ;pointer to ethernet
                                   ;Table
        mov     bx, [bx]        ;pointer to structure

        cmp     filter_id, 1     ;always filter_id
        jz      short @msd_cont1 ; = 1 is master

;        cmp     edx, dword ptr [bx].address
;        jb     short @msd_cont1
;        ja     short @msd_cont3
;        cmp     bp, word ptr [bx+4].address
;        jbe     short @msd_cont1

@msd_cont3:
        inc     di

@msd_cont1:
        loop    @msd_cont0

@msd_cont2:
        mov     ax, di
        pop     ds
        ret

MasterSlaveDecision endp

```

comment -

```

=====

PlayerInTeam counts number of players in Blue Team and Red Team.
Input: ax = number of players-1
       es:[di] = table to store ethernet address and team mode
              (minimum size = MAXNUMBEROFMIGS * TeamEntryLength
              + 2)
Output: al = number of players in Blue Team
        ah = number of players in Red Team
        table --> number of entries -- word
              entries -- 8 bytes each

=====

```

```

PlayerInTeam  proc  far
;
        push    ds
        push    di
        mov     cx, ax

        xor     ax, ax
        xor     bx, bx
        xor     dx, dx
        add     di, 2

```

```

@pit_cont0:
    push    ax
    push    bx
    push    cx
    push    dx
    mov     ax, cx
    dec     ax
    call    far ptr RcvSome
    or      ax, ax
    pop     dx
    pop     cx
    pop     bx
    pop     ax
    jnz     short @pit_cont1      ;error

    cmp     word ptr [si+6], MODEO_LENGTH
    jnz     short @pit_cont0
    cmp     [si+10], bx          ;status 0 -
                                ;decision
    jnz     short @pit_cont0
    cmp     byte ptr [si+12], RedTeam
    jnz     short @pit_cont2
    inc     ah
    jmp     short @pit_cont3

@pit_cont2:
    cmp     byte ptr [si+12], BlueTeam    ; n o t
                                            ;necessary
    inc     al

@pit_cont3:
    movsd                     ;ethernet
                                ;address
    movsw
    add     si, 6
    movsb
    inc     di
    inc     dx

@pit_cont1:
    loop    @pit_cont0

    pop     di
    mov     es:[di], dx
    pop     ds
    ret

PlayerInTeam endp

```

comment -

```
=====
SortEthernetAddress sorts ethernet address to get plane ID number.
Input: ds:[si] = table created by PlayerInTeam
       es:[di] = table of ID # in sort of ascending order
           (enemies -- in order, followed by friends in
            order)
       dl = team mode
       dh = master
Output: table is sorted based on ethernet address
=====
```

```
SortEthernetAddress    proc    far
;
                        push    di
                        push    si

                        push    si
                        lodsw
                        mov     cx, ax
                        shl     cx, 3                      ;times 8
                                                           ;TeamEntryLength=
8
                        mov     bx, si
                        add     bx, cx

                        mov     ebp, dword ptr ethernetaddress
                        mov     [bx], ebp
                        mov     bp, ethernetaddress[4]
                        mov     [bx+4], bp
                        mov     [bx+6], dl

                        push    di
                        push    dx

@sea_cont1:
                        mov     di, si
                        add     di, TeamEntryLength
                        mov     cx, ax
                        mov     ebx, [si]
                        mov     edx, [si+4]

@sea_cont2:
                        cmp     ebx, [di]
                        jb      short @sea_cont3
                        ja      short @sea_cont4            ;swap
                        cmp     dx, [di+4]
                        jb      short @sea_cont3
```

```

@sea_cont4:      xchg     ebx, [di]
                  xchg     edx, [di+4]

@sea_cont3:      add      di, TeamEntryLength
                  loop     @sea_cont2

                  mov      [si], ebx
                  mov      [si+4], edx
                  add      si, TeamEntryLength
                  dec      ax
                  jnz      short @sea_cont1

                  pop      dx
                  pop      di
                  pop      si

                  lodsw
                  inc      ax
                  push     si
                  mov      bp, ax
                  mov      cx, ax
;save counter

@sea_cont6:      add      si, 6
;skip ethernet
;address

                  lodsw
                  cmp      al, dl
                  jz       short @sea_cont5

                  mov      ax, bp
                  sub      ax, cx
                  cmp      al, dh
                  jz       short @sea_cont5
                  stosb

@sea_cont5:      loop     @sea_cont6

                  pop      si
                  mov      cx, bp

@sea_cont7:      add      si, 6
                  lodsw
                  cmp      al, dl
                  jnz      short @sea_cont8

                  mov      ax, bp
                  sub      ax, cx
                  cmp      al, dh
                  jz       short @sea_cont8
                  stosb

```

```

@sea_cont8:      loop      @sea_cont7

                  mov      byte ptr es:[di], -1

                  pop      si
                  pop      di
                  push     es
                  mov      ax, cs
                  mov      es, ax

                  lodsw
                  mov      cx, ax
                  mov      bp, ax                ;number of nodes
                                                  ;in the network
                  inc      cx

@sea_cont0:      lodsd
                  mov      edx, eax
                  lodsd
                  cmp      edx, dword ptr ethernetaddress
                  jnz      short @sea_cont9
                  cmp      ax, ethernetaddress[4]
                  jz       short @sea_cont10

@sea_cont9:      push     si
                  push     bp
                  push     cx
                  push     di
                  mov      cx, bp
                  mov      bx, offset code:ethernetIndex

@sea_cont11:     mov      di, es:[bx]
                  mov      di, es:[di]
                  cmp      edx, dword ptr es:[di].address
                  jnz      short @sea_cont12
                  cmp      ax, word ptr es:[di+4].address
                  jnz      short @sea_cont12

                  mov      bp, sp
                  mov      si, [bp]
                  mov      al, [si]
                  cbw
                  mov      es:[di].planeID, ax
                  jmp      short @sea_cont13

@sea_cont12:     add      bx, 2
                  loop     @sea_cont11

```

```

@sea_cont13:
        pop     di
        pop     cx
        pop     bp
        pop     si
        inc     di

@sea_cont10:
        loop    @sea_cont0

        pop     es
        ret

SortEthernetAddress    endp

; -----
;      ExitRcvInt
; -----
ExitRcvInt  proc      near

        ired

ExitRcvInt  endp

; -----
savvecs  proc      near
        push    ds
        push    es
        push    si
        push    di
        push    cx

        mov     ax,ds
        mov     es,ax
        xor     ax,ax
        mov     ds,ax
        mov     cx,22h           ;vectors 0 - 21h, 2 wds per
        mov     di,offset CODE:vectsv
        xor     si,si
        cld
        cli
        rep     movsd           ;save 'em all
        sti

        pop     cx
        pop     di
        pop     si
        pop     es
        pop     ds
        ret
savvecs  endp

```

```

;-----
fixvecs proc      near
    push    es
    push    si
    push    di
    push    cx

    xor     ax,ax
    mov     es,ax
    mov     cx,22h      ;vectors 0 - 21h, 2 wds per
    mov     si,offset CODE:vectsv
    xor     di,di
    cld
    cli
    rep     movsd      ;restore 'em all
    sti

    pop     cx
    pop     di
    pop     si
    pop     es
    ret
fixvecs endp

RCODE     ENDS
END

```

12.3 F3ltoC.asm

title f3ltoC.asm

```
*****
;
; File: F3LTOC.ASM
;
; Description: This file contains subroutines which provide the
;              C program with an interface to the 3L 1.0 routines.
;              This program prepared two receiving buffers, one
;              for ASAT1 and the other for ASAT2. The ASAT
;              addresses are replaced by the PC's address. Each
;              received packet is time-stamped.
;
; > rcvptrq+2  2              4
;              +-----+-----+
;              | packet address | packet length |
;              +-----+-----+
; rcvptrq 0-1 0              1
;              +-----+
;              | number packets |
;              +-----+
;
; The Host memory buffers are ring queues.
;
*****

; Functions called by C programs

PUBLIC  _getds

PUBLIC  _cInitParameters
PUBLIC  _cInitAdapters
PUBLIC  _cResetAdapter
PUBLIC  _cWhoAmI
PUBLIC  _cRdRxFilter
PUBLIC  _cWrRxFilter
PUBLIC  _cSetLookAhead
PUBLIC  _etext
```


PUBLIC _cXmit1

PUBLIC _cInitBufPtr

PUBLIC _cGet_Parmsptr

PUBLIC _cGetPkttrxPtr

PUBLIC _cRcvFlag

PUBLIC _cGetNumPkt

PUBLIC _cGetOnePkt

PUBLIC _cPassHead

PUBLIC _cmyAddress

PUBLIC _cGetTimeCount

PUBLIC _cGettimeptr

PUBLIC _cGetTimeQPtr

;Need to be written in C

extrn _myExitRcvInt :near

;not used in this version

extrn _myRxProcess :near

;not used in this version

extrn _myTxProcess :near

;not used in this version

extrn _CheckHead :near

;Functions provide by this file

PUBLIC ExitRcvInt

PUBLIC RxProcess

PUBLIC TxProcess

;not used in this version

;3L library functions

extrn InitParameters :near

extrn InitAdapters :near

extrn WhoAmI :near

extrn ResetAdapter :near

extrn RdRxFilter :near

extrn WrRxFilter :near

extrn GetRxData :near

extrn SetLookAhead :near

extrn PutTxData :near

;3L 503 library variables

extrn v_hdr_size :word

extrn packet_hdr_addr :word

;define constants

lf equ 0ah
cr equ 0dh
Hdlen equ 024h
PacketLength equ 566
Qbytes equ 13584 ;buffer size per Asat
MaxPktInQ equ Qbytes/PacketLength

CODE GROUP _TEXT, DATA, ICODE

_TEXT segment byte public 'CODE'
DGROUP group _DATA, _BSS
assume cs:_TEXT, ds:DGROUP, ss:DGROUP
_TEXT ends

DATA segment word public 'CODE'
DATA ends

ICODE segment word public 'CODE'
ICODE ends

DATA segment

his_ds dw ?
his_es dw ?
int_ds dw ?
int_es dw ?
int_di dw ?
int_si dw ?
int_cx dw ?
int_dx dw ?
_etext db ?

```
;WhoAmI adapter info structure
```

```
ad_info struc
```

```
ea      db      6 dup(0)      ;enet addr
ver1    db      0              ;major ver
ver2    db      0              ;minor ver
ver3    db      0              ;sub ver
ver4    db      0              ;type ver
atyp    db      0              ;adapter type
astat   db      0              ;adapter status
bfrs    db      0              ;buffer flags
nxb     db      0              ;number of xmit buffers
sxb     dw      0              ;xmit buffer size
xmtc    dd      0              ;xmit count
xmte    dd      0              ;xmit errs
xmtto   dd      0              ;xmit timeouts
rcvc    dd      0              ;rcv count
rcvbc   dd      0              ;bcast rcv count
rcve    dd      0              ;rcv errs
rtc     dd      0              ;retry count
xfmd    db      0              ;xfer mode flags
wtmd    db      0              ;wait mode flags
extp    dw      0              ;extension pointer
ad_info ends
```

```
id_struc
```

```
struc
```

```
address  db      6 dup(0)      ;ethernet address
count    db      0              ;byte to check packet ID (whether
                                ;the same packet received twice
headPointer dw      0
tailPointer dw      0
headAddress dw      0          ;head pointer to pktData
tailAddress dw      0          ;tail pointer to pktData
pktData   db      Qbytes dup(0)
timehead  dw      0
timetail  dw      0
timeIN    dd      MaxPktInQ dup(0) ;store time stamp at
                                ;packet in
timenext  dw      0            ;pointer of next time
                                ;address to
id_struc  ends                ;timetail
```

```
;WhoAmI buffer
```

```
wbf      ad_info <>          ;WhoAmI
buffer
myaddress db      02h,60h,8ch,4bh,0dh,9bh ;pc 3COM
address
rcvflag   dw      0            ;0: disable
packet rcv.                                ;1: enable
```

```
packet rcv.
```

```
;Asat buffers
```

```
Asat1     id_struc <>
Asat2     id_struc <>
```

```
ttailptr  dw      0            ; current time stamping pointer
tnextptr  dw      0            ; next time stamping pointer
```

```

stkcheck      dw      0ABCDh      ; stack clobber check dw
              dw      512 dup(0)
topstack      dw      0           ; adapter 0 stack top (and stack
                                   ; in use flag)

vectsv  dd      22h dup (0)      ;save all vectors so we can
                                   ;cleanup
retsav  dw      ?
crlf    db      cr,lf,'$'

pklen   dw      0
pkerr   dw      0
pkcnt   dw      0
pkcount dw      0
asatid  dw      0                ;asat id
addid   dw      0                ;

trxbuf  db      PacketLength dup(0) ;buffer for transmitting
pkthd   db      36 dup(0)

; time variables

temp_lo      db      0
temp_hi_bit  db      0
timelo       dw      0
timehi       dw      0
temp_lo1     db      0
temp_hi_bit1 db      0
timelo1      dw      0
timehi1      dw      0
nextlo       dw      0
nexthi       dw      0

DATA        ends

_DATA      segment word public 'DATA'
_d@        label   byte
_DATA      ends
_BSS       segment word public 'BSS'
_b@        label   byte
_BSS       ends
_DATA      segment word public 'DATA'
_s@        label   byte
_DATA      ends

_TEXT      SEGMENT
          ASSUME  CS:_TEXT, DS:DGROUP, SS:DGROUP

```

```

_getds  proc      near
        push      bx
        mov       ax,ds
        mov       cs:his_ds,ax
        mov       ax,es
        mov       cs:his_es,ax
        mov       bx, seg Asat1
        pop       bx
        ret
_getds  endp

```

```

;-----
;_cGet_Parmsptr : This subroutine returns a pointer points to the
;                  address of WhoAmI structure 'wbf' and MaxPktInQ.
; Calling sequence:
;   cGet_Parmsptr(&parmsdr,&TotPktsInQ);
;
; INPUT : Non
; Return: Non
;-----

```

```

_cGet_Parmsptr  proc  near
                push  bp
                mov   bp,sp
                push  si
                push  di
                push  bx
                push  ds

                mov   ax,cs
                mov   ds,ax

                mov   bx, offset code:wbf           ; get WhoAmI address
                pop   ds
                mov   si,[bp+4]
;               mov   word ptr [si+2],ax           ; pass segment cs
                mov   word ptr [si],bx             ; pass pointer

                mov   bx, MaxPktInQ
                mov   si, [bp+6]
                mov   word ptr [si], bx           ; pass queue size

                xor   ax, ax

                pop   bx
                pop   di
                pop   si
                pop   bp
                ret
_cGet_Parmsptr  endp

```

```

;-----
;_cmyAddress
; This subroutine sets up 'myaddress' from wbf.address. It is
; called after InitAdapters.
;
; Calling sequence:
;   cmyAddress();
;
; Return: NON
;-----
_cmyAddress  proc    near
              push    ds
              push    cx

              mov     ax, cs
              mov     ds, ax

              mov     cx, 6
              mov     si, offset code:wbf          ; source address
              mov     di, offset code:myaddress    ; destination
                                                    ; address
              rep     movsb

              xor     ax, ax

              pop     cx
              pop     ds
              ret
_cmyAddress  endp

```

```

;-----
;_cRcvFlag
; This subroutine sets up the receiving flag 'rcvflag' which is
; used by RxProcess to start to receive packets if rcvflag = 1.
;
; Calling sequence:
;   cRcvFlag(Rcvflag);
;
; Return: NON
;-----
_cRcvFlag    proc    near
              push    bp
              mov     bp,sp
              push    ds

              mov     ax,cs
              mov     ds,ax

              mov     ax, [bp+4]          ; get receive flag
              mov     cs:rcvflag, ax      ; save flag

              xor     ax, ax

              pop     ds
              pop     bp
              ret
_cRcvFlag    endp

```

```

;-----
;_cGettimeptr : This subroutine returns the time pointer points at
;               low word to C program.
;Calling sequence:
;               cGettimeptr(&timeptr)
;Return: Non
;-----
_cGettimeptr    proc    near

                push    bp
                mov     bp,sp
                push    si
                push    ds

                mov     ax,cs

                pop     ds
                mov     si,[bp+4]
                mov     word ptr [si],offset cs:timelo
                mov     word ptr [si+2],ax

                pop     si
                pop     bp
                ret

_cGettimeptr    endp

```



```

;-----
; GetTimeCount
;   This function returns a timestamp constructed of the Timer
;   0 value and the lowest word of the MS-DOS clock. The Timer
;   0 is a count-down timer, so it is converted to form a
;   coherent timestamp value. The Timer value is returned in
;   the AX register (low word) and the clock value is returned
;   in the DX register (high word).
;-----

```

```

GetTimeCount    proc    near
                 push    ds                ;set segment pointer for
                                     ;clock read
                 push    dx
                 push    ax

                 mov     ax,0040h        ;
                 mov     ds,ax            ;

                 mov     al,0c2h         ;set up for count/status latch

;                 cli                ;no ints here
                 out     043h,al         ;latch
                 mov     dx,ds:006ch     ;get clock lsw
;                 sti                ;restore ints

                 mov     cs:timehi,dx    ;store time high word

                 in      al,040h         ;get status
                 and     al,080h         ;get msbit
                 mov     cs:temp_hi_bit,al ;store msbit
                 in      al,040h         ;get lsb of count
                 mov     cs:temp_lo,al   ;store lsb of count
                 in      al,040h         ;get msb of count
                 mov     ah,al
                 mov     al,cs:temp_lo   ;get count into ax reg
                 ror     ax,1
                 or      ah,cs:temp_hi_bit ;get back bit 16
                 not     ax              ;change from
                                     ;count-down to
                                     ;count-up

                 mov     cs:timelo,ax    ;store time low word

                 pop     ax
                 pop     dx
                 pop     ds              ;restore segment
                                     ;pointer

                 ret

GetTimeCount    endp

```

```

;-----
; GetTimeCount1
;   This function returns a timestamp constructed of the Timer
;   0 value and the lowest word of the MS-DOS clock. The Timer
;   0 is a count-down timer, so it is converted to form a
;   coherent timestamp value. The Timer value is returned in
;   the AX register (low word) and the clock value is returned
;   in the DX register (high word).
;-----

GetTimeCount1    proc    near
                  push    ds                ;set segment pointer for
                                          ;clock read
                  push    dx
                  push    ax

                  mov     ax,0040h        ;
                  mov     ds,ax            ;

                  mov     al,0c2h          ;set up for count/status
                                          ;latch

;                  cli                ;no ints here
;                  out     043h,al        ;latch
;                  mov     dx,ds:006ch    ;get clock lsw
;                  sti                ;restore ints

                  mov     cs:timehi1,dx    ;store time high word

                  in     al,040h           ;get status
                  and     al,080h         ;get msbit
                  mov     cs:temp_hi_bit1,al ;store msbit
                  in     al,040h         ;get lsb of count
                  mov     cs:temp_lo1,al   ;store lsb of count
                  in     al,040h         ;get msb of count
                  mov     ah,al
                  mov     al,cs:temp_lo1   ;get count into ax
                                          ;reg
                  ror     ax,1
                  or      ah,cs:temp_hi_bit1 ;get back bit 16
                  not     ax              ;change from
                                          ;count-down
                                          ;to count-up

                  mov     cs:timelol,ax    ;store time low
                                          ;word

                  pop     ax
                  pop     dx
                  pop     ds              ;restore segment
                                          ;pointer

                  ret

GetTimeCount1    endp

```

```

;-----
; cGetTimeCount
;     This subroutine is prepared for C program call
;     GetTimeCount.
;-----

_cGetTimeCount    proc    near
                    call    GetTimeCount
                    ret
_cGetTimeCount    endp

;-----
;_cGetTimeQPtr : This subroutine returns the time buffer tail
;                pointer and the next pointer, if it is available,
;                to the C program.  AsatID decides the timer Q.
;Calling sequence:
;                cGetTimeQPtr(AsatID,&TimePtr,&NextTimePtr);
;Return: AX : 0
;           1 - error
;-----
_cGetTimeQPtr proc    near
                    push    bp
                    mov     bp,sp
                    push    si
                    push    di
                    push    cx
                    push    bx
                    push    ds

                    mov     ax, cs
                    mov     ds, ax

                    mov     bx, [bp+4]                ; get asat ID

                    cmp     bx, 1
                    jnz     @gettime1
                    mov     di, offset cs:Asat1        ; load asat1
                                                         ; address
                    jmp     short @gettime2
@gettime1:
                    cmp     bx, 2
                    jnz     short @gettimeE
                    mov     di, offset cs:Asat2        ; load asat2
                                                         ; address
@gettime2:
                    mov     cx, [di].timetail
                    mov     cs:ttailptr, cx
                    mov     cx, [di].timenext
                    mov     cs:tnextptr, cx

                    pop     ds
                    mov     si, [bp+6]
                    mov     cx, cs:ttailptr
                    mov     word ptr [si], cx
                    mov     word ptr [si+2], ax

```

```

        mov     si, [bp+8]
        mov     cx, cs:tnextptr
        mov     word ptr [si], cx
        mov     word ptr [si+2], ax

        xor     ax, ax
        jmp     short @_gettimeEx

@_gettimeE:
        pop     ds
        mov     ax, 1

@_gettimeEx:

        pop     bx
        pop     cx
        pop     di
        pop     si
        pop     bp
        ret

_cGetTimeQPtr endp

```

```

;-----
;
;_cInitAdapters:      This procedure provides the glue between a C
;                      program and the 3L 1.0 InitAdapters function.
;
;Calling Sequence:
;    int cInitAdapters(&nAdapters)
;
;Input Parameters:
;    None
;
;Output Parameters:
;    int nAdapters
;
;Returns:
;    The return value of the InitAdapters function
;-----

```

```

_cInitAdapters proc near
    push     bp
    mov     bp, sp
    push     si
    push     di
    push     ds

    mov     ax, cs
    mov     ds, ax
    mov     di, offset CODE:RxProcess

```

```

        call    InitAdapters

        pop     ds
        mov     di,word ptr[bp+4]
        mov     word ptr[di],cx

        pop     di
        pop     si
        pop     bp
        ret
_cInitAdapters endp

;-----
;
;_cInitParameters:  This procedure provides the glue between a C
;                   program and the 3L 1.0 InitAdapters function.
;
;Calling Sequence:
;   int cInitParameters(Parms)
;
;Input Parameters:
;   char *Parms - Pointer to a structure with overrides of default
;                parameters.
;
;Output Parameters:
;   None
;
;Returns:
;   The return value of the InitParameters function
;-----
_cInitParameters proc near
        push    bp
        mov     bp,sp
        push    si
        push    di
        push    ds

        mov     bx,[bp+4]
        mov     ax,ds
        mov     es,ax
        mov     ax,cs
        mov     ds,ax

        call    savvecs
        call    InitParameters
        pop     ds
        pop     di
        pop     si
        pop     bp
        ret
_cInitParameters endp

```

```

;-----
;
;_cResetAdapter:  This procedure provides the glue between a C
;                  program and the 3L 1.0 ResetAdapters function.
;
;Calling Sequence:
;    int cResetAdapter()
;
;Input Parameters:
;    None
;
;Output Parameters:
;    None
;
;Returns:
;    The return value of the ResetAdapter function
;
;-----
_cResetAdapter proc near
    push    bp
    mov     bp,sp
    push    si
    push    di
    push    ds

    mov     dx,0
    mov     ax,cs
    mov     ds,ax

    mov     dl,0
    call    ResetAdapter
    call    fixvecs

    pop     ds
    pop     di
    pop     si
    pop     bp

    ret
_cResetAdapter endp

```

```

;-----
;
;_cWhoAmI:  This procedure provides the glue between a C
;           program and the 3L 1.0 WhoAmI function.
;
;Calling Sequence:
;   int cWhoAmI(&WhoPtr)
;
;Input Parameters:
;   None
;
;Output Parameters:
;   struct WhoStruct far *WhoPtr - Far pointer to the WhoAmI
structure
;
;Returns:
;   The return value of the WhoAmI function
;
;-----

```

```

_cWhoAmI proc near
    push    bp
    mov     bp,sp
    push    si
    push    di
    push    ds

    mov     dx,0
    mov     ax,cs
    mov     ds,ax

    call    WhoAmI

    pop     ds
    mov     si,[bp+4]
    mov     Word ptr [si],di
    mov     Word ptr [si+2],es

    pop     di
    pop     si
    pop     bp
    ret
_cWhoAmI endp

```

```

;-----
;
;_cRdRxFilter:   This procedure provides the glue between a C
;                program and the 3L 1.0 RdRxFilter function.
;
;Calling Sequence:
;    int cRdRxFilter(&RxFilter)
;
;Input Parameters:
;    None
;
;Output Parameters:
;    int RxFilter - The receive filter value
;
;Returns:
;    The return value of the RdRxFilter function
;
;-----
_cRdRxFilter proc near
    push    bp
    mov     bp,sp
    push    si
    push    di
    push    ds

    mov     ax,cs
    mov     ds,ax

    mov     dx,0
    call    RdRxFilter

    pop     ds
    mov     di,[bp+4]
    mov     [di],bx

    pop     di
    pop     si
    pop     bp
    ret
_cRdRxFilter endp

```



```

;-----
;
;_cWrRxFilter:   This procedure provides the glue between a C
;                program and the 3L 1.0 WrRxFilter function.
;
;Calling Sequence:
;   int cWrRxFilter(RxFilter)
;
;Input Parameters:
;   int RxFilter - The new receive filter value
;
;Output Parameters:
;   None
;
;Returns:
;   The return value of the WrRxFilter function
;
;-----
_cWrRxFilter proc near
    push    bp
    mov     bp,sp
    push    ds
    push    si
    push    di

    mov     ax,cs
    mov     ds,ax

    mov     dx,0
    mov     ax,[bp+4]
    call    WrRxFilter

    pop     di
    pop     si
    pop     ds
    pop     bp
    ret
_cWrRxFilter endp

```

```

;-----
;
;_cSetLookAhead:   This procedure provides the glue between a C
;                  program and the 3L 1.0 SetLookAhead function.
;
;Calling Sequence:
;    int cSetLookAhead(NumBytes)
;
;Input Parameters:
;    int NumBytes - The number of bytes of look ahead data
;
;Output Parameters:
;    None
;
;Returns:
;    The return value of the SetLookAhead function
;
;-----
_cSetLookAhead proc near
    push    bp
    mov     bp,sp
    push    si
    push    di
    push    ds

    mov     ax,cs
    mov     ds,ax

    mov     dx,0
    mov     ax,[bp+4]
    call    SetLookAhead

    pop     ds
    pop     di
    pop     si
    pop     bp
    ret
_cSetLookAhead endp

```

```

;-----
;
;TxProcess:  This procedure is the protocol-side routine which is
;             called when a packet has finished transmitting (see
;             cInitAdapters).  It provides the glue between the 3L
;             1.0 routines and C routine called myTxProcess.
;
;myTxProcess Calling Sequence:
;  void myTxProcess(Status, RequestID)
;
;myTxProcess Input Parameters:
;  int Status - Receive status
;  int RequestID - The request identifier
;
;myTxProcess Returns:
;  Nothing
;
;-----

```

```

TxProcess proc near
    push    bp
    push    si
    push    di
    push    ds
    push    es

    push    ax
    mov     ax,cs:his_ds
    mov     ds,ax
    mov     es,ax
    pop     ax

    xor     cx,cx
    mov     cl,dh
    xor     dh,dh

    push    cx
    push    ax
    call    _myTxProcess

    add     sp,4

    pop     es
    pop     ds
    pop     di
    pop     si
    pop     bp
    ret
TxProcess endp

```

```

;-----
;
;ExitRcvInt: This procedure is the protocol-side routine which is
;             called when the 3L has completed a receive interrupt.
;-----

```

```
ExitRcvInt proc near
```

```
    irect
```

```
ExitRcvInt endp
```

```

;-----
;_cPassHead: This subroutine should be called by 'C' program at
;             least once after the call to '_cInitBufPtr' in order
;             to pass the address of 'pkthd' to 'Hdptr->inh' in 'C'.
;Calling sequence:
;             cPassHead(&Hdptr);
;Return: NON
;-----

```

```

_cPassHead    proc    near
               push    bp
               mov     bp,sp
               push    si
               push    bx
               push    es
               push    ds

               mov     ax,cs
               mov     ds,ax

               pop     ds
               mov     si,[bp+4]
               mov     word ptr [si],offset cs:pkthd+4
               mov     word ptr [si+2],ax

               pop     es
               pop     bx
               pop     si
               pop     bp
               ret
_cPassHead    endp

```

```

;-----
;_cGetPkttrxPtr : This subroutine returns the transmitting buffer
;                  pointer to the C program.
;Calling sequence:
;                  cGetPkttrxPtr(&Pkttrxptr)
;Return: Non
;-----

```

```

_cGetPkttrxPtr proc    near

```

```

        push    bp
        mov     bp,sp
        push    si
        push    ds

        mov     ax,cs

        pop     ds
        mov     si,[bp+4]
        mov     word ptr [si],offset cs:trxbuf
        mov     word ptr [si+2],ax

        pop     si
        pop     bp
        ret

```

```

_cGetPkttrxPtr endp

```

```

;-----
; get_bufHdptr:  Get the available buffer header address in
;                  Asatx.pktData
; INPUT          : bx = Asat ID, 1 for Asat1, 2 for Asat2.
; OUTPUT         : di = Asatx.headAddress
;-----

```

```

get_bufHdptr proc    near
        push    ds
        push    ax

        mov     ax, cs
        mov     ds, ax

        cmp     bx, 1
        jnz     @gethptr1
        mov     di, offset code:Asat1
        mov     di, [di].headAddress
        jmp     short @gethptr2

```

```

@gethptr1:      cmp     bx, 2
                 jnz     @gethptr2
                 mov     di, offset code:Asat2
                 mov     di, [di].headAddress

```

```

@gethptr2:      pop     ax
                 pop     ds
                 ret

```

```

get_bufHdptr endp

```

```

;-----
;  update_Asathdptr:  Update  Asatx.count,  Asatx.headPointer,
;                    Asatx.headAddress after receiving a packet.
; INPUT      :  bx = Asat ID, 1 for Asat1, 2 for Asat2.
; OUTPUT     :  NON
;-----

```

```

update_Asathdptr  proc  near
                  push  ds
                  push  di
                  push  ax

```

```

                  mov   ax, cs
                  mov   ds, ax

```

```

                  cmp   bx, 1
                  jnz   @updateHd1
                  mov   di, offset code:Asat1
                  jmp   short @updateHd2

```

```

@updateHd1:      cmp   bx, 2
                  jnz   @updateHdE
                  mov   di, offset code:Asat2

```

```

@updateHd2:      cmp   [di].headPointer, MaxPktInQ
                  jnz   @updateHd3
                  mov   [di].headPointer, 0
                  mov   ax, di
                  add   ax, 15
                  sub   ax, PacketLength
                  mov   [di].headAddress, ax
                  mov   ax, di
                  add   ax, 15+Qbytes
                  mov   [di].timehead, ax

```

```

@updateHd3:      inc   [di].count
                  inc   [di].headPointer
                  add   [di].headAddress, PacketLength
                  add   [di].timehead, 4

```

@updateHdE:

```
    pop    ax
    pop    di
    pop    ds
    ret
```

update_AsathDptr endp

```
-----
; PutTime: Put time in timelo/timehi into Asatx.timeIN pointed by
;           Asatx.timehead.
; INPUT     : bx = Asat ID, 1 for Asat1, 2 for Asat2.
; OUTPUT    : NON
;-----
```

```
PutTime    proc    near
            push    di
            push    ax

            cmp     bx, 1
            jnz     @puttime1
            mov     di, offset code:Asat1
            jmp     short @puttime2

@puttime1:  cmp     bx, 2
            jnz     @puttimeE
            mov     di, offset code:Asat2

@puttime2:  mov     di, [di].timehead
            mov     ax, cs:timelol
            mov     [di], ax
            mov     ax, cs:timehil
            mov     2[di], ax

@puttimeE:  pop     ax
            pop     di
            ret

PutTime    endp
```

```

;-----
;
;RxProcess:      This procedure is the protocol-side routine which is
;                  called when a packet has been received (see
;                  cInitAdapters). It provides the glue between the
;                  3L 1.0 routines and C routine called myRxProcess.
;
;myRxProcess Calling Sequence:
;  void myRxProcess(Status, PacketSize, RequestID, PacketHeader)
;
;myRxProcess Input Parameters:
;  int Status - Receive status
;  int PacketSize - Size of the received packet
;  int RequestID - The request identifier
;  char far *PacketHeader - Address of the virtual packet header
;
;myRxProcess Returns:
;  Nothing
;-----

```

```

RxProcess proc near
    push    bp
    push    di
    push    si
    push    ds
    push    es
    push    bx

    mov     ax,cs
    mov     ds,ax

    mov     cs:pklen,cx                ; save packet length
    mov     ax,cs:rcvflag              ; get flag
    cmp     ax,0                       ; if 1, receive packet
    jz      @Rxcnt0                    ; release 3com buffer

    mov     cs:int_dx,dx
    mov     cs:int_cx,cx
    mov     cs:int_ds,ds
    mov     cs:int_es,es
    mov     cs:int_di,di
    mov     cs:int_si,si
    mov     ds,cs:his_ds
    mov     es,cs:his_es

    call    _CheckHead

```



```

        mov     ds,cs:int_ds
        mov     es,cs:int_es
        mov     di,cs:int_di
        mov     si,cs:int_si
        mov     cx,cs:int_cx
        mov     dx,cs:int_dx

        mov     cs:asatid, ax
        cmp     ax, 1
        jz      @Rxcnt1
        cmp     ax, 2
        jz      @Rxcnt2
@Rxcnt0:
        xor     cx, cx                                ; packet length 0
                                                    ; release adapter buffer
;         mov     di,offset CODE:pktdat ; load dummy buffer address
        jmp     @Rxcnt5
@Rxcnt1:
        mov     di, offset code:Asat1
        xor     ax, ax
        mov     al, [di].count                        ; get packet count
        cmp     ax, MaxPktInQ
        jz      @Rxcnt0
        mov     cx, cs:pklen                          ; restore packet length
        mov     bx, 1
        call    get_bufHdptr                          ; get header address from
        jmp     short @Rxcnt5                        ; Asat1

@Rxcnt2:
        mov     di, offset code:Asat2
        xor     ax, ax
        mov     al, [di].count                        ; get packet count
        cmp     ax, MaxPktInQ
        jz      @Rxcnt0
        mov     cx, cs:pklen                          ; restore packet length
        mov     bx, 2
        call    get_bufHdptr                          ; get header address from
                                                    ; Asat2

@Rxcnt5:
        mov     cs:pkerr,0
;         mov     di,offset CODE:pkthd ;buffer
;;        mov     di,cs:bufptr ;load offset in the buffer
        or      dl,40h                                ; release buffer

```

```

; *****
call    GetRxData
; *****
jcxz    @Rxcnt6
mov     cs:pkerr,ax
mov     cs:pklen,cx

mov     bx, cs:asatid           ; get asat ID
call    GetTimeCount1         ; get system time
call    PutTime                ; put time in the time Q
call    update_AsatHdptr       ; update Asat head pointer
inc     cs:pkcount

@Rxcnt6:
pop     bx
pop     es
pop     ds
pop     si
pop     di
pop     bp
ret

RxProcess endp

;-----
; get_bufTlptr:      Get the available buffer tail address in
;                   Asatx.pktData
; INPUT             : bx = Asat ID, 1 for Asat1, 2 for Asat2.
; OUTPUT            : si = Asatx.tailAddress
;-----
get_bufTlptr proc    near
    push    ds
    push    ax

    mov     ax, cs
    mov     ds, ax

    cmp     bx, 1
    jnz     @gettptr1
    mov     si, offset code:Asat1
    mov     si, [si].tailAddress
    jmp     short @gettptr2

@gettptr1:
    cmp     bx, 2
    jnz     short @gettptr2
    mov     si, offset code:Asat2
    mov     si, [si].tailAddress

@gettptr2:
    pop     ax
    pop     ds
    ret

get_bufTlptr endp

```

```

;-----
; renew_address: Replaces the PC's address to Asat address at the
;                 first 6 bytes.
; INPUT  : si - pointer to the packet.
; OUTPUT : si - pointer to the packet.
;-----

```

```

renew_address proc    near
    push    es
    push    ds
    push    si
    push    di
    push    cx

    mov     cx, cs
    mov     es, cx
    mov     ds, cx

    mov     cx, 6
    mov     di, si
    mov     si, offset code:myaddress      ; destination
    rep     movsb                          ; source

    pop     cx
    pop     di
    pop     si
    pop     ds
    pop     es
    ret
renew_address endp

```

```

;-----
; update_AsatTlptr: Update Asatx.count, Asatx.tailPointer,
;                   Asatx.tailAddress after transmitting a packet.
; INPUT   : bx = Asat ID, 1 for Asat1, 2 for Asat2.
; OUTPUT  : NON
;-----

```

```

update_AsatTlptr proc    near
    push    ds
    push    di
    push    ax

    mov     ax, cs
    mov     ds, ax

    cmp     bx, 1
    jnz     @updateTl1
    mov     di, offset code:Asat1
    jmp     short @updateTl2

```

```

@updateTl1:      cmp     bx, 2
                  jnz     @updateTlE
                  mov     di, offset code:Asat2

@updateTl2:      cmp     [di].tailPointer, MaxPktInQ
                  jnz     short @updateTl3
                  mov     [di].tailPointer, 0
                  mov     ax, di
                  add     ax, 15
                  sub     ax, PacketLength
                  mov     [di].tailAddress, ax
                  mov     ax, di
                  add     ax, 15+Qbytes
                  mov     [di].timetail, ax

@updateTl3:      dec     [di].count
                  inc     [di].tailPointer
                  add     [di].tailAddress, PacketLength
                  add     [di].timetail, 4
                  cmp     [di].tailPointer, MaxPktInQ
                  jnz     short @updateTl4
                  mov     ax, di
                  add     ax, 15+Qbytes
                  mov     [di].timenext, ax

@updateTl4:      add     [di].timenext, 4

@updateTlE:      pop     ax
                  pop     di
                  pop     ds
                  ret

update_Asatlptr endp

```

```

;-----
; _cXmit1: Transmits one packet either from Asat1.pktData or
;           Asat2.pktData based on the ASAT ID passed from
;           filter.c. This subroutine also replaces the ASAT address
;           with the PC's address.
; Calling sequence:
;           _cXmit1(ttlpl, nb, flags, reqid, pktptr, &nreqid, AsatID,
;           newaddflag);
; INPUT :   ttlpl,nb = packet length.
;           AsatID = 0, trx. packet from pointer pktptr, otherwise
;                   from Asatx.tailAddress.
;           newaddflag = 0, no address replacement. 1, replace PC's
;                   address (AsatID > 0).
; Return:   Non
;-----
; transmit one packet
_cXmit1    proc    near
    push    bp
    mov     bp,sp
    push    si
    push    di
    push    ds

    mov     ax,ds
    mov     es,ax

    ;setup for PutTxData

    mov     cx, [bp+6]                ; set lengths
    mov     dl, byte ptr[bp+8]
    mov     dh, byte ptr[bp+10]
    mov     bx, [bp+18]                ; get Asat ID

    mov     ax, [bp+20]                ; get new address ID

    mov     cs:asatid, bx               ; save Asat ID
    mov     cs:addid, ax               ; save Address ID

    cmp     bx, 0
    jz      @Xtrx1

    mov     ax, cs
    mov     ds, ax
    mov     es, ax

    call    get_bufTlptr                ; set si=Asatx.tailAddress

    mov     ax, cs:addid                ; restore address ID
    cmp     ax, 0
    jz      short @Xtrx2

    call    renew_address                ; replace a new adapter
address    jmp     short @Xtrx2

```

```

@Xtrx1:
    mov     ax, cs
    mov     ds, ax

    mov     si, [bp+12]
;    mov     si, offset cs:trxbuf

@Xtrx2:
    mov     bx, [bp+4]                ; set packet lengths
    mov     di, 0ffffh               ; no TxProcess

    call    PutTxData                 ; transfer data to the
                                      ; adapter buffer

    pop     ds
    xchg    dh, dl
    xor     dh, dh
    mov     di, [bp+16]
    mov     [di], dx

    mov     bx, cs:asatid             ; restore Asat ID
    cmp     bx, 0
    jz      @Xtrx3
    call    update_Asatlptr

@Xtrx3:
    pop     di
    pop     si
    pop     bp
    ret

_cXmit1   endp

```

```

;-----
; InitBufptr
; This subroutine initializes the receiving buffer pointers and
; counters.
;-----
InitBufptr   proc   near

    mov     di, offset code:Asatl     ; load Asatl address
    mov     word ptr [di].count, 0    ; reset counter
    mov     word ptr [di].headPointer, 1 ; initialize
                                      ; pointers
    mov     word ptr [di].tailPointer, 1 ;
    mov     bx, di
    add     bx, 15
    mov     word ptr [di].headAddress, bx ; store address
                                      ; Asatl.pktData
    mov     word ptr [di].tailAddress, bx ; store address
                                      ; Asatl.pktData
    add     bx, Qbytes+4               ; load time
                                      ; buffer address

    mov     [di].timehead, bx
    mov     [di].timetail, bx
    mov     [di].timenext, bx
    add     [di].timenext, 4

```

```

;-----
;_cGetNumPkt
; This subroutine returns a number of total packets in the
; receiving buffer.
; Calling sequence:
;   number = cGetNumPkt(AsatID);
;
; INPUT : Asat ID - 1 for Asat1, 2 for Asat2
; Return: AX - number of packets in the buffer
;-----
_cGetNumPkt    proc    near
                push    bp
                mov     bp,sp
                push    ds
                push    di

                mov     ax,cs
                mov     ds,ax

                mov     ax, [bp+4]                ; get Asat ID
                cmp     ax, 1
                jnz     @getnum1
                mov     di, offset code:Asat1
                xor     ax, ax
                mov     al, [di].count            ; store a number
into AX

                jmp     short @getnum3

@getnum1:       cmp     ax, 2
                jnz     @getnum2
                mov     di, offset code:Asat2
                xor     ax, ax
                mov     al, [di].count            ; store a number
into AX

                jmp     short @getnum3

@getnum2:       xor     ax, ax                    ; error, return
0
@getnum3:       pop     di
                pop     ds
                pop     bp
                ret
_cGetNumPkt    endp

```

```

        mov     ax, word ptr 6[bx]           ; get packet length
        mov     bx, [bp+8]                   ; get clear flag
        cmp     bx, 0
        jz      short @getpkt4
        mov     bx, cs:asatid                 ; get Asat ID
        call    update_AsatTlptr
        inc     cs:pkcnt

@getpkt4:

        pop     ds
        pop     bx
        pop     di
        pop     si
        pop     bp
        ret

_cGetOnePkt endp

;-----
savvecs proc     near
        push    ds
        push    es
        push    si
        push    di
        push    cx

        mov     ax, ds
        mov     es, ax
        xor     ax, ax
        mov     ds, ax
        mov     cx, 22h*2                    ;vectors 0 - 21h, 2 wds per
        mov     di, offset CODE:vectsv
        xor     si, si
        cld
        cli
rep     movsw                                ;save 'em all
        sti

        pop     cx
        pop     di
        pop     si
        pop     es
        pop     ds
        ret
savvecs endp

```


13.0 Data_acq.c

This is a stand alone program used to capture ASAT simulator data.

```
/******  
Data_acq.c
```

This program samples and stores the five channel data from ASAT simulator.

```
*****
```

```
#include <stdio.h>  
#include <stdlib.h>  
#include <conio.h>  
#include <time.h>  
#include <dos.h>  
#include <bios.h>  
#include <math.h>  
#include "kklib.h"  
#include "clib.h"
```

```
/* Program to sample and store data using mode 27 */
```

```
#define DSIZE      10                /* DAS20 Data Array Size  
                                     (Arguments) */  
#define ASIZE      655              /* Array for Conditioned  
                                     Data */  
#define BUFSIZE    32750            /* # samples in a buffer */  
#define SCANSIZE    5  
#define INITVALUE  0                /* Value to initialize  
                                     darray to */
```

```

long      noc,nos;
int       error, nom, gcount;
int       i, n;
long      FileSizePerMinute = 55000;
long      GlobalNumOfMinutes;
int far *buffer;
int far *buffer1;
int far *buffer2;
int far *bufptr;
int far *ALLOC();
char      *timec;
char      *FileToUse;
char      *LogFile;
char      *Subject;
char      *Condition;
char      *Task;
char      *VTRStart;
char      *VTREnd;
char      *Bernoulli;
long      datafilesize, logfilesize;
FILE      *fin;
extern int LastKey;

```

```

/*-----*/
char *DetermineVTREnd()
{
    int      shr, smin, ssec, sfr;
    double   ehr = 0;
    double   efr = 0;
    double   emin = 0;
    double   esec = 0;
    double   temp, tempd;
    char      dummy[3];

    strncpy(dummy, VTRStart, 2);
    dummy[2] = '\0';

    shr = atof(dummy);
    strncpy(dummy, VTRStart+3, 2);
    dummy[2] = '\0';

    smin = atof(dummy);
    strncpy(dummy, VTRStart+6, 2);
    dummy[2] = '\0';

    ssec = atof(dummy);
    strncpy(dummy, VTRStart+9, 2);
    dummy[2] = '\0';
}

```

```

sfr = atof(dummy);

temp = noc / 5;
temp = temp / 1000;
temp = modf(temp, &emin);
temp = temp * 60;
temp = modf(temp, &esec);
efr = ceil(temp * 30);
efr += sfr;

if (efr > 29)
{
    efr -= 30;
    esec++;
}

esec += ssec;

if (esec > 59)
{
    esec -= 60;
    emin++;
}

emin += smin;

if (emin > 59)
{
    emin -= 60;
    ehr++;
}

ehr = ehr + shr;

itoa((int) ehr, VTREnd, 10);
strcat(VTREnd, ":");
itoa((int) emin, dummy, 10);
strcat(VTREnd, dummy);
strcat(VTREnd, ":");
itoa((int) esec, dummy, 10);
strcat(VTREnd, dummy);
strcat(VTREnd, ":");
itoa((int) efr, dummy, 10);
strcat(VTREnd, dummy);
return(VTREnd);
}

```

```

    if ((GlobalNumOfMinutes * FileSizePerMinute) >= avail)
    {
        ErrorMessage("Not enough disk space for file.");
        return(False);
    }

    if (access(filename, 0) != 0)
        return(True);
    else
    {
        Question(    "File already exists. Overwrite (Y/N)? ",
                    "N", &ans[0], CYAN, WHITE);

        return(strcmp(ans, "Y") == 0);
    }
}

/*-----*/
BOOLEAN ValidMinutes(char mins[])
{
    int i = atoi(mins);

    GlobalNumOfMinutes = i;          /* Used in ValidFileName to
                                     calculate required disk size of
                                     output file. */

    return((i > 0) && (i < 19));
}

/*-----*/
BOOLEAN IsBernoulliActive()
{
    char buffer[512];
    int  result;

    printf("\nTesting to see if drive M: is ready\n");
    result = biosdisk(16,138,0,0,0,1,buffer);
    result &= 0xAA;

    return(result);
}

```

```

/*-----*/
void Init(void)
{
    int x = 0;

    Attr(BLUE, WHITE);
    clrscr();
    GetVideoMode();
    DrawTopBar("ASAT Flight Stick Data Acquisition System");
    DrawBottomBar(25, "~Esc~-Exit");

    for (x = 1; x < 81; x++)
        PokeChar(x, 8, 'M', (BLUE << 4) | WHITE);

    for (x = 1; x < 81; x++)
        PokeChar(x, 21, 'M', (BLUE << 4) | WHITE);

    Attr(GREEN, WHITE);
    PokeStr(32, 8, " LOG INFORMATION ");
    PokeStr(29, 21, " EXECUTION INFORMATION ");

    timec = (char *) malloc(15);
    timec[14] = '\0';

    if ((buffer1 = ALLOC(32768)) == NULL) /* Allocate buffer */
    {
        ErrorMessage("Cannot allocate buffer #1. Aborting
                     program.");
        exit(0);
    }

    if ((buffer2 = ALLOC(32768)) == NULL) /* Allocate buffer */
    {
        ErrorMessage("Cannot allocate buffer #2. Aborting
                     program.");
        exit(0);
    }
}

```

```
FileToUse = (char *) malloc(41);
FileToUse[0] = '\0';
LogFile = (char *) malloc(41);
LogFile[0] = '\0';
Subject = (char *) malloc(31);
Subject[0] = '\0';
Condition = (char *) malloc(9);
Condition[0] = '\0';
Task = (char *) malloc(31);
Task[0] = '\0';
VTRStart = (char *) malloc(12);
VTRStart[0] = '\0';
VTREnd = (char *) malloc(12);
VTREnd[0] = '\0';
Bernoulli = (char *) malloc(2);
Bernoulli[0] = 'Y';
Bernoulli[1] = '\0';
```

}

```

/*-----*/

void DoReads()
{
    struct GetHead    G;
    char              nomstr[5] = "    \0";

    InitGet(&G);
    AddGet(&G, 5, 4, 4, "Enter Number of Minutes to Run (1-7):",
&nomstr[0], Numeric, 2, False);
    ValidateGet(G.Last, ValidMinutes);
    AddGet(&G, 5, 6, 40, "Enter File Name:", FileToUse,
AlphaNumeric, 1, False);
    ValidateGet(G.Last, ValidFileName);
    AddGet(&G, 5, 10, 30, "Subject(1-th;2-fl;3-ch;4-ca):", Subject,
AlphaNumeric, 3, False);
    AddGet(&G, 5, 12, 8, "Condition:", Condition, AlphaNumeric, 4,
False);
    AddGet(&G, 30, 12, 30, "Task:", Task, AlphaNumeric, 5, False);
    AddGet(&G, 5, 14, 11, "VTR Counter Start:", VTRStart, TimeCode,
6, False);
    AddGet(&G, 5, 16, 1, "Backup to Bernoulli:",
        Bernoulli, AlphaNumeric, 7, False);
    Get(G);
    FreeGets(G);
    nom = atoi(nomstr);
    Attr(BLUE, WHITE);
    FlushKeybdBuffer();
    if (LastKey == EscKey)
    {
        CursorSmall();
        Attr(BLACK, WHITE);
        clrscr();
        exit(0);
    }
    return;
}

```

```

/*-----
char *TimeStr(unsigned int timenum)
{
    int min, sec;
    char dummy[7];

    min = timenum / 9000;
    sec = (timenum / 150) % 60;
    itoa(min, timec, 10);
    itoa(sec, dummy, 10);
    timec = (char *) strcat(timec, ":");
    timec = (char *) strcat(timec, dummy);
    timec = (char *) PadLeft(timec, 14);
    return(timec);
}

/*-----
void _initdas() /* This function will initialize the DAS-20 */
{
    int mode0[10];

    /* Initialize DAS-20 Board */
    /* Initialization Mode */
    mode0[0] = 0x350; /* Base Address */
    mode0[1] = 2; /* Interrupt Level */
    mode0[2] = 1; /* DMA Level */

    if ((error = DAS20(0, mode0)) != 0) /* CALL DAS20 */
    {
        printf("\nMode 0 Error = %d. Aborting program.",error);
        exit(0);
    }
}

/*-----*/
void _loadq() /* This function will load the queue with our info */
{
    int model[DSIZE];
    int chanel[DSIZE];

```



```

mode27[0] = nos1;          /* # of scans */
mode27[1] = SEGADR(buffer1); /* Paragraph Address of Buffer */
mode27[2] = 0;             /* Specifies External Trigger */
mode27[3] = 1;             /* Specifies Stop at End of
                           samples */

if ((error = DAS20(27,mode27)) != 0) /* CALL DAS20 */
{
    printf("\nMode 27 Error = %d. Aborting Program.",error);
    exit(0);
}

PokeStr(5, 22, "Time:");
PokeStr(45, 22, "Conversion #:");

mode27[1] = 1;
while (mode27[1] != 0)
{
    /* Monitor Status */
    if ((error = DAS20(12,mode27)) != 0)
    {
        printf("\nMode 12 Error = %d. Aborting
                Program.",error);
        exit(0);
    }
    PokeStr(10, 22, TimeStr(mode27[2]));
    ltoa(mode27[2], dummy, 10);
    PokeStr(59, 22, dummy);

    if (kbhit())
    {
        InKey();
        if (LastKey == EscKey)
        {
            noc = mode27[2];
            DAS20(11, mode27);
            break;
        }
    }
}

}

/*-----*/

int _sample2() /* This function will take samples with an
               external */
/* Clock and stop at the end of conversions */
{
    int nos2;
    unsigned int mode27[DSIZE];
    char dummy[7];

    nos = nom * 1800;
    noc = 5 * nos; /* Number of Conversions */

    if (nos > (int) (BUFSIZE/SCANSIZE))
        nos2 = (int) nos-(int) (BUFSIZE/SCANSIZE);
}

```

```

/*-----*/
void _filedat(unsigned int dindex)
{
    int          darray[ASIZE];
    int          model3[DSIZE];
    int          i,j,flag;
    unsigned int bufindex;
    unsigned int index;
    char         input[20];

    /* Condition Data */
    for(i = 0; i < ASIZE; i++)
        darray[i] = INITVALUE;

    if ((dindex + ASIZE) <= noc)
        j = ASIZE;
    else
        j = noc-dindex;

    if (dindex >= BUFSIZE) {
        bufptr = buffer2;
        bufindex = dindex-BUFSIZE;
    }
    else {
        bufptr = buffer1;
        bufindex = dindex;
    }

    model3[0] = j;
    model3[1] = SEGADR(bufptr);
    model3[2] = bufindex;

    model3[3] = OFFADR(darray);
    model3[4] = -1;
    model3[5] = 1;
    model3[6] = 0;

    /* # of Conversions */
    /* Raw Data Buffer */
    /* Starting at offset into */
    /* buffer */
    /* Data Array */
    /* No Channel Array */
    /* Bipolar */
    /* SSH-4 not used. */

    if ((error = DAS20(13, model3)) != 0) /* CALL DAS20 */
    {
        printf("\nMode 13 Error = %d. Aborting Program.",error);
        exit(0);
    }

    for (i = 0; i < j; i+=5) {
        gcount++;
        fprintf(fin, " %d %d %d %d %d %d\n",gcount, darray[i],
darray[i+1], darray[i+2],darray[i+3],darray[i+4]);
    }
}

```

```

if (Bernoulli[0] == 'Y' || Bernoulli[0] == 'y')
{
    getdfree(13, &free);
    avail = (long) free.df_avail *
            (long) free.df_bsec * (long) free.df_sclus;

    if ((logfilesize + datafilesize) >= avail)
    {
        ErrorMessage("Not enough disk space available on the
                     Bernoulli.");
        return;
    }

    PokeStr(22, 24, "Copying information to Bernoulli...");
    gotoxy(1, 23);
    dummy[0] = '\0';
    strcpy(dummy, "COPY ");
    strcat(dummy, FileToUse);
    strcat(dummy, " M:\\\\BACKUP\\");
    strcat(dummy, FileToUse);
    system(dummy);
    PokeStr(1, 23, "                                ");
    gotoxy(1, 23);
    strcpy(dummy, "COPY ");
    strcat(dummy, LogFile);
    strcat(dummy, " M:\\\\BACKUP\\");
    strcat(dummy, LogFile);
    system(dummy);
    PokeStr(1, 23, "                                ");
}
gotoxy(1, 24);
clreol();
gotoxy(1, 22);
clreol();
return;
}

```

```

PokeStr(26, 24, "Saving information to disk...");
fin = fopen(FileToUse, "w");

if ((noc % ASIZE) == 0)
    count = noc / ASIZE;
else
    count = noc / ASIZE + 1;

gcount = 0;

for (i = 0; i < count; i++)
    _filedat(i*ASIZE);

datafilesize = ftell(fin);
fclose(fin);
DumpLog();
}
while (True);
}

```

14.0 Stripcha.c

```
/******  
STRIPCHA.C
```

This program shows a stripchart of horizontal and vertical stick positions , throttle position, flare, chaff, cage/uncage missile and gun fire events.

```
*****/
```

```
#include <stdio.h>  
#include <stdlib.h>  
#include <alloc.h>  
#include <string.h>  
#include <conio.h>  
#include <graphics.h>  
#include <dos.h>  
#include <time.h>  
#include <process.h>
```

```
char    *path = "j:\\tc\\bgi\\";  
char far *timec;  
char    dummy[15];  
int huge *buffer,*bufferstart;  
int      min,sec,frame, option;  
long int count,countstart;
```

```
/*-----/  
This program draws the horizontal and vertical bars and sets the  
c o l o r s .  
-----/
```

```

void setimage()
{
    int i,y1,y2,y3,y4,y5;
    clearviewport();
    setbkcolor(0);
    setcolor(63);
    line(150,10,150,190);          /* Draw the horizontal and */
    line(50,100,600,100);          /* vertical coordinates    */
    line(50,200,600,200);
    setcolor(14);
    outtextxy(65,1,"Horizontal");
    outtextxy(75,56,"Left");
    outtextxy(75,140,"Right");
    setcolor(10);
    outtextxy(160,1,"Vertical");
    outtextxy(480,56,"Front");
    outtextxy(480,140,"Rear");
    setcolor(63);
    outtextxy(550,110,"Time");      /* Show the time coordinate */
    outtextxy(300,110,"5s");
    outtextxy(450,110,"10s");
    outtextxy(140,80,"2");
    outtextxy(140,62,"4");
    outtextxy(140,44,"6");
    outtextxy(140,26,"8");
    outtextxy(132,115,"-2");
    outtextxy(132,133,"-4");
    outtextxy(132,151,"-6");
    outtextxy(132,169,"-8");

    line(150,210,150,290);
    line(50,290,600,290);
    outtextxy(140,272,"2");
    outtextxy(140,256,"4");
    outtextxy(140,240,"6");
    outtextxy(140,224,"8");

    setcolor(9);

    switch(option)                  /* Determine which option is selected */
    {
        case 1: outtextxy(80,210,"Throttle"); break;
        case 2: outtextxy(80,210,"Flair");   break;
        case 3: outtextxy(80,210,"Chaff");    break;
        case 4: outtextxy(80,210,"Cage");      break;
                outtextxy(80,280,"Uncage");    break;
    }
}

```

```

setcolor(63);
outtextxy(550,300,"Time");
outtextxy(300,300,"5s");
outtextxy(450,300,"10s");
setcolor(14);
outtextxy(550,192,"Missile");          /* Show the missile
                                         coordinate */

setcolor(4);
outtextxy(550,202,"Gun");              /* Show machine-gun
                                         coordinate */

setcolor(12);

for(i=1;i<=10;i++)
{
    line(150,(100+9*(i-1)),450,(100+9*(i-1)));
}

for(i=12;i<=21;i++)
{
    line(150,(100+9*(i-1)),450,(100+9*(i-1)));
}

for(i=0;i<=9;i++)
{
    line(150,(250+8*(i-5)),450,(250+8*(i-5)));
}

for(i=1;i<=11;i++)
{
    line((150+30*(i-1)),10,(150+30*(i-1)),190);
    line((150+30*(i-1)),210,(150+30*(i-1)),290);
}

setcolor(63);                          /* Set color "white" */
outtextxy(100,300,"Time:");
outtextxy(150,300,timec);              /* Display time */
line(50,315,600,315);
outtextxy(150,330,"Pause ---- Press [Esc]");

for(i=150;i<450;i++)                  /* Display the four channel
                                         signals */
{
    y1 = 0.05*(*(buffer+6*(i-150)+1))+ 65;
    y2 = -0.05*(*(buffer+6*(i-150)+2)) + 138;
    y3 = *(buffer+6*(i-150)+3);
    y4 = *(buffer+6*(i-150)+4);
    y5 = *(buffer+6*(i-150)+5);
    putpixel(i, y1 , 14 );
    putpixel(i, y2 ,10 );
}

```

```

        if(y4<500)
        {
            setcolor(14);
            line(i,195,i,200);
        }

        if(y5<500)
        {
            setcolor(4);
            line(i,200,i,205);
        }

        switch(option) /* Display the selected channel signal */
        {
            case 1: putpixel(i, -0.054*y3 + 300, 9); break;

            case 2:
            case 3:
            {
                if (y3 < 500)
                {
                    setcolor(9);
                    line(i,210,i,290);
                }
                else
                    putpixel(i,290,9);

                break;
            }

            case 4: putpixel(i, y3, 9); break;
        }
    }
}

/*-----
/* This function generates the time string format mm:ss:frame
/*-----

void timestring(void)
{
    char dummy1[17],dummy2[2];

    dummy2[0] = ':';
    dummy2[1] = '\0';
    frame = frame + 1;
}

```



```

do                /* Receive the first digit of minute */
{
    word = getch();
    word = word - 48;

    if((word>=0) && (word<=9))
    {
        min = min + 10*word ;
        itoa(word,wordstr,10);
        outtextxy(150,340,wordstr);
    }
}
while((word<0) || (word>9));

do                /* Receive the second digit of minute */
{
    word = getch();
    word = word - 48;

    if((word>=0) && (word<=9))
    {
        min = min + word;
        itoa(word,wordstr,10);
        outtextxy(160,340,wordstr);
    }
}
while((word<0) || (word>9));

do                /* Receive the first digit of second */
{
    word = getch();
    word = word - 48;

    if((word>=0) && (word<=9))
    {
        sec = sec + 10*word;
        itoa(word,wordstr,10);
        outtextxy(172,340,wordstr);
    }
}
while((word<0) || (word>9));

```

```
        if(count<0)
        {
            setviewport(150,340,220,349,0);
            clearviewport();
            setviewport(0,0,639,349,0);
            outtextxy(10,340,"Reenter Time");
        }
    }
    while( count<0);
}
```

```

/*-----
/* This function shows titles on the screen
/*-----

void init_screen(void)
{
    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            "SA reverse",
            NULL);

    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            "WINDOW 2, 10, 6, 69, BOLD CYAN ON BLUE SHADOW ZOOM",
            NULL);

    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            " ROWCOL 3,21 "
BLUE",
            "' AVIATION RESEARCH LABORATORY' BOLD WHITE ON
            NULL);

    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            " ROWCOL 5,21 "
BLUE",
            "' *****INTERSIMULATOR NETWORK*****' BOLD WHITE ON
            NULL);
}

```

```

/*-----
/* This function accepts the data file name
/*-----

void wait_load(char pt_file[])
{
    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            " WINDOW 10,10,14,69 BOLD CYAN ON BLUE SHADOW ZOOM",
            NULL);

    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            " ROWCOL 12,17 'Please enter data file name :' BOLD
WHITE ON BLUE",
            NULL);

    scanf("%s",pt_file);
}

/*-----
/* This function loads the data and related side information
/*-----

void end_load(char pt_file[])
{
    int    i, j, flag, status;
    char    cha,subj,pt_log[70];
    FILE    *fp;

    struct entry
    {
        char content[70];
    }
    line [20];

    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            " WINDOW 8,10,23,69 BOLD CYAN ON BLUE SHADOW ZOOM",
            NULL);

    i = 0;

    while ( pt_file[i] != '.' && pt_file[i] != '\0' )
    {
        pt_log[i] = pt_file[i];
        ++i;
    }
}

```

```

pt_log[i] = '.';
pt_log[i+1] = 'l';
pt_log[i+2] = 'o';
pt_log[i+3] = 'g';
pt_log[i+4] = '\\0';

fp = fopen(pt_log,"r");
i = 0;
flag = 1;

while (flag != EOF && i < 9)
{
    j = 0;
    cha = 1;

    while(cha != '\\n')
    {
        flag = fscanf(fp,"%c",&cha);
        line[i].content[j] = cha;
        ++j;
    }

    line[i].content[j] = '\\0';

    if (line[i].content[0] != '\\n')
    {
        gotoxy(18,11+i);
        printf("%s",line[i].content);
        i++;
    }
}

subj = line[2].content[31];
switch(subj)          /* Determine which channel is selected */
{
    case '1' : option = 1; break;
    case '2' : option = 2; break;
    case '3' : option = 3; break;
    case '4' : option = 4; break;
}

```

```

    if(option == 4)
    {
        status = 1;                /* Uncage */

        for(i=0;i<countstart;i=i+6)
        {
            if ( *(bufferstart + 3 + i+ 6) - *(bufferstart + 3 + i)
< -500 )
                status = 1/ (status + 1) ;

            if(status == 1)
                *(bufferstart + 3 + i) = 290;                /*low*/
            else
                *(bufferstart + 3 + i ) = 210;                /*high*/
        }
    }

    spawnl( P_WAIT,
            "BE.EXE",
            "BE",
            " ROWCOL 21,17 'Press any key to continue' BOLD WHITE
ON BLUE",
            NULL);

    i = getch();
    fclose(fp);
}

/*-----
/* This is the main function, It sets the graphic mode,
/* initializes all the parts and controls the display process.
/*-----
void main(void)
{
    int driver,mode,k,pagenum,flag,word1,word2;
    int intin;
    FILE *fd;
    char name[70];

    if((buffer = (int huge *) farmalloc (98302*sizeof(int)))
==NULL)
    {
        printf("Memory allocation error\n");
        exit(0);
    }
    /* Allocate data memory */

    bufferstart = buffer;

    init_screen();

```

```

do
{
    wait_load(name);
    fd =fopen(name, "r");
}
while(fd == NULL);

timec = dummy;
count = 0;

do
/* Read data file */
{
    flag = fscanf(fd,"%d", &intin);

    if(flag ==EOF) break;
    *buffer = intin;

    buffer = buffer + 1;
    count = count +1;
}
while(flag != EOF );

buffer = bufferstart;
countstart = count;

end_load(name);

driver = VGA;
mode = VGAMED;
initgraph(&driver,&mode,path);

setactivepage(0);
/* Set active video memory */

min = 0;
sec = 0;
frame = 0;
timestring();
setimage();
setvisualpage(0);

pagenum=1;

```

```

do
{
    buffer = buffer + 6;
    count = count - 6;
    timestring();
    setactivepage(pagenum);
    setimage();
    setvisualpage(pagenum);
    pagenum = (pagenum + 1) % 2;      /* Switch active page and
                                     visual page */

    if(kbhit())
    {
        word1 = getch();

        if(word1 == 27)
        {
            setcolor(0);
            outtextxy(150,330,"Pause ---- Press [Esc]");
            setcolor(63);
            outtextxy(150,330, "Replay --- Press [Enter];  "
                      "Exit --- Press [Esc];");

            outtextxy(150,340,"Play from specific time ---
Press [P]");

            do
            {
                word2 = getch();

                /* Process 'Esc', 'Enter' or 'P' selection*/

                switch(word2)
                {
                    case 27:
                    {
                        restorecrtmode();
                        clrscr();
                        exit(0);
                    }

                    case 112:
                    {
                        setstarttime();
                        break;
                    }

                }
            }
            while( (word2 != 13) && (word2 != 27) && (word2
!=112) );
        }
    }
    while( count > 0);

    restorecrtmode();

```



```
    clrscr();  
    return(0);  
}
```

15.0 Volt3.c

```
/*-----  
Use this program to translate data files generated from the Keithly  
A/D board (in packed binary format) into voltages. The output of  
the file will be as follows:
```

```
\ Comment  
4 3  
0.033330 0.863760 -0.893040  
0.066660 0.922320 -0.844240  
0.099990 -1.000400 0.922320  
0.133320 1.464000 1.171200
```

The first line represents the comment field. This field may be up to (but not exceeding) 4 lines. The second line represents the number of rows and columns in the file (in this case 4 rows of 3 columns). The next four lines represent the data collected. The first column is the time at which the data were collected. The second and third columns are the X and Y channels respectively (X and Y stick position values).

```
-----
```

```
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>
```

```
char input[20];
```

```
char *filein; /* Asat data FILE */  
char *fileout;  
FILE *fout, *fin;
```

```
char *outfiles[9];  
char *files[7];  
int num, rc = 0;  
int frame, second, min, hour;  
long offset = 0;
```

```
/*-----  
void Comment(int l)  
{  
    char *comment;  
    int i = 0;  
    int len;  
  
    comment = (char *) malloc(80);  
    offset = 0;
```

```

clrscr();
gotoxy(1,1);
printf ("Enter comment for output file (4 lines max)\n");

if(l == 0)
    comment = gets(comment);

while (i < 4)
{
    comment[0] = '\0';

    printf("%ld. ",i+1);
    gets(comment);
    len = strlen(comment);
    offset += len;

    if (len == 0)
        i = 4;
    else
    {
        if (i == 0)
            offset += 3;
        else
            offset += 4;
        i++;
        fprintf(fout, "\\ %s\n",comment);
    }
}

fprintf(fout, "$                \n");
offset += 1;
}

```

```
/*-----  
void GetFiles(void)  
{  
    int i, j;  
  
    for (i=0; i<7; i++)  
        files[i] = (char *) malloc(10);  
  
    printf ("Enter number of files to transform\n");  
    scanf ("%d", &num);  
  
    printf ( "Enter names of files to transform\n");  
    for(j = 0; j < num; j++)  
        scanf ("%s", files[j]);  
}
```

```

/*-----
void LineCount(int lc)
{
    int    ccount = 3;
    char  ch, line1[80];

    fout = fopen(fileout, "r+");
    fseek(fout, offset, SEEK_SET);

    fprintf(fout, "%d    %d", lc, ccount);

    rewind(fout);

    while (fgets(line1, 80, fout));
        fclose(fout);
}

/*-----
void TimeCode(void)
{
    frame +=1;

    if (frame == 30)
    {
        frame = 0;
        second += 1;

        if(second == 60)
        {
            second = 0;
            min += 1;

            if(min == 60)
            {
                min = 0;
                hour += 1;

                if(hour == 24)
                    hour = 0;
            }
        }
    }
}

```

```

        switch (j)
        {
            case 0 :
            {
                strcpy(hrstr, temp);
                hrstr[2] = '\\0';
                break;
            }

            case 1 :
            {
                strcpy(minstr, temp);
                minstr[2] = '\\0';
                break;
            }

            case 2 :
            {
                strcpy(secstr, temp);
                secstr[2] = '\\0';
                break;
            }

            case 3 :
            {
                strcpy(frstr, temp);
                frstr[2] = '\\0';
                break;
            }
        }

        j += 1;
    }

    frame = -1;
    second = atoi(secstr);
    min = atoi(minstr);
    hour = atoi(hrstr);
}

/*-----
void main(void)
{
    int k, l, temp = 0;
    float volt0, volt1, temp0, temp1, sec = 0;
    int print;

```

```

clrscr();
gotoxy(1,1);
GetFiles();
OutFiles();

printf("Choose the format which you wish \n");
printf("to print to %s\n", fileout);
printf("1. = X and Y channels only\n");
printf("2. = Time in milliseconds X and Y channels\n");
printf("3. = Time in milisec. X and Y channels and Time
code\n");
scanf("%d", &print);

if(print == 3)
    InitTime();

filein = (char *) malloc(20);
fileout = (char *) malloc(20);

for (k = 0; k< num; k++)
{
    filein = files[k];
    fin = fopen(filein, "r");
    l = k;

    if (k == 0)
    {
        fileout = outfiles[k];
        fout = fopen(fileout, "w");
        Comment(1);
    }

    rc =

    while(!feof(fin))
    {
        if (temp == 12600 && !feof(fin))
        {
            fclose(fout);
            LineCount(temp);
            temp = 0;
            fileout = outfiles[l];
            fout = fopen(fileout, "w");
            Comment(1);
            l += 1;
        }
    }
}

```

```

fscanf (fin, "%f %f\n" , &temp0, &temp1);
rc++;
temp++;
sec += .03333;
volt0 = temp0 * .00488;
volt1 = temp1 * .00488;

switch(print)
{
    case 1 :
    {
        if (volt0 < 0 && volt1 < 0)
            fprintf (fout, "%6f %6f\n", volt0, volt1);
        else
            if (volt0 >= 0 && volt1 >= 0)
                fprintf (fout, "%6f %6f\n", volt0,
volt1);
            else
                if (volt0 >= 0 && volt1 < 0)
                    fprintf (fout, "%6f %6f\n", volt0, volt1);
                else
                    if (volt0 < 0 && volt1 >= 0)
                        fprintf (fout, "%6f %6f\n", volt0,
volt1);
                    gotoxy (5,24);
                    printf ("Volt0 = %6f Volt1 = %6f", volt0,
volt1);
                    break;
            }
        case 2 :
        {
            if (volt0 < 0 && volt1 < 0)
                fprintf (fout, "%f %6f %6f\n", sec,
volt0, volt1);
            else
                if (volt0 >= 0 && volt1 >= 0)
                    fprintf (fout, "%f %6f %6f\n", sec,
volt0, volt1);
                else
                    if (volt0 >= 0 && volt1 < 0)
                        fprintf (fout, "%f %6f %6f\n", sec,
volt0, volt1);
                    else
                        if (volt0 < 0 && volt1 >= 0)
                            fprintf (fout, "%f %6f %6f\n", sec,
volt0, volt1);
                        gotoxy (5,24);
                        printf ("Volt0 = %6f Volt1 = %6f", volt0,
volt1);
                        break;
                    }
    }
}

```



```

        case 3 :
        {
            TimeCode();

            if(volt0 < 0 && volt1 < 0)
                fprintf (fout, "%f      %6f      %6f
%d:%d:%d:%d\n",
sec, volt0, volt1, hour, min, second, frame);
            else
                if(volt0 >= 0 && volt1 >= 0)
                    fprintf (fout, "%f      %6f      %6f
%d:%d:%d:%d\n",
sec, volt0, volt1, hour, min, second, frame);
            else
                if(volt0 >= 0 && volt1 < 0)
                    fprintf (fout, "%f      %6f      %6f
%d:%d:%d:%d\n",
sec, volt0, volt1, hour, min, second, frame);
            else
                if(volt0 < 0 && volt1 >= 0)
                    fprintf (fout, "%f      %6f      %6f
%d:%d:%d:%d\n",
sec, volt0, volt1, hour, min, second, frame);

            gotoxy (5,24);
            printf ("Volt0 = %6f      Volt1 = %6f", volt0,
volt1);
            break;
        }
    }

    if (eof(fin) && temp == 12600)
    {
        fclose(fout);
        LineCount(temp);
    }

    if (eof(fin) && k == num -1)
    {
        fclose(fout);
        LineCount(temp);
    }

    fclose(fin);
}

exit(0);
)

```